Treebank-Based Grammar Acquisition for German

Ines Rehbein

A dissertation submitted in fulfilment of the requirements
for the award of
Doctor of Philosophy (Ph.D.)

to the

Dublin City University

School of Computing

Supervisor: Prof. Josef van Genabith

August 2009

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy (Ph.D.) is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed _____ (Ines Rehbein)
Student ID: 5513 0917
Date: May 2009

Acknowledgements

I wish to acknowledge everyone who has helped me with this thesis. First of all, I would like to thank my supervisor, Josef van Genabith, who's unshakeable optimism and constant believe that my time at DCU will result in a publishable piece of research never ceased to surprise me. He was right, after all...

Thanks to my fellow students in the GramLab project, Amine Akrout, Gregorz Chrupała, Yvette Graham, Yuqing Guo, Masanori Oya and Natalie Schluter, for their support and interest in my work. Special thanks to Yuqing, who has been a great friend, and who never got tired of joining in my complaints over the Irish weather.

I'd also like to thank other past and present members of the NCLT, Ríona Finn, Jennifer Foster, Deirdre Hogan, Sara Morrisey, Karolina Owczarzak, Lamia Tounsi and Joachim Wagner. Thanks for many inspiring chats during lunch and coffee breaks, revitalising jogs in the park, and for giving encouragement when I needed it.

I am especially grateful to Jennifer Foster and Sandra Kübler for many useful comments on my thesis.

My life in Dublin, especially during the first year, would not have been the same without my friends. I'd like to thank Susanne Lechle for strenuous hikes in the mountains, relaxing walks on the beach, shared cooking experiences and great nights out in the pub. Dublin would have been less fun without her!

I also want to thank my collegues in Saarbrücken, Caroline Sporleder and Josef Ruppenhofer, for encouraging and supporting me while I was writing up my thesis. I'm very fortunate to work with them.

Finally, I would like to express my gratitude to the Science Foundation Ireland who supported my research with grant 04/IN/I527.

Contents

List of Figures

Abstract

Manual development of deep linguistic resources is time-consuming and costly and therefore often described as a bottleneck for traditional rule-based NLP. In my PhD thesis I present a treebank-based method for the automatic acquisition of LFG resources for German. The method automatically creates deep and rich linguistic representations from labelled data (treebanks) and can be applied to large data sets.

My research is based on and substantially extends previous work on automatically acquiring wide-coverage, deep, constraint-based grammatical resources from the English Penn-II treebank (Cahill et al., 2002; Burke et al., 2004b; Cahill, 2004). Best results for English show a dependency f-score of 82.73% (Cahill et al., 2008) against the PARC 700 dependency bank, outperforming the best hand-crafted grammar of Kaplan et al. (2004). Preliminary work has been carried out to test the approach on languages other than English, providing proof of concept for the applicability of the method (Cahill et al., 2003; Cahill, 2004; Cahill et al., 2005).

While first results have been promising, a number of important research questions have been raised. The original approach presented first in Cahill et al. (2002) is strongly tailored to English and the data-structures provided by the Penn-II treebank (Marcus et al., 1993). English is configurational and rather poor in inflectional forms. German, by contrast, features semi-free word order and a much richer morphology. Furthermore, treebanks for German differ considerably from the Penn-II treebank as regards data structures and encoding schemes underlying the grammar acquisition task.

In my thesis I examine the impact of language-specific properties of German and of linguistically motivated treebank design decisions on PCFG parsing and LFG grammar acquisition. I present experiments investigating the influence of treebank design on PCFG parsing and show which type of representations are useful for the PCFG and LFG grammar acquisition task. Furthermore I present a novel approach for cross-treebank comparison, measuring the effect of controlled error insertion on treebank trees and parser output from different treebanks. I complement the cross-treebank comparison by augmenting a human evaluation on the TePaCoC, a new testsuite for testing parser performance on complex grammatical constructions. The manual evaluation on the TePaCoC provides new insights on the impact of flat vs. hierarchical annotation schemes on data-driven parsing. In my thesis I present treebank-based LFG acquisition methodologies for two German treebanks. An extensive evaluation along different dimensions complements the investigation and provides valuable insights for the future development of treebanks.

Chapter 1

Introduction

Over the last two decades, deep wide-coverage linguistic resources such as grammars have attracted interest from different areas in NLP. Deep linguistic resources can provide useful information for NLP applications such as Information Retrieval, Question Answering, Information Extraction or Machine Translation. Typically, deep linguistic resources are hand-crafted. Unfortunately, the development of hand-crafted deep, wide-coverage linguistic resources is extremely time-consuming, knowledge-intensive and expensive. Many hand-crafted resources are domain-dependent and exhibit a serious lack of coverage. Therefore, more and more attention has been focused on data-driven methods for the automatic acquisition of linguistic resources, mostly in the area of data-driven grammar acquisition or automatic acquisition of lexical resources (Sharman et al., 1990; Brent, 1991, 1993; Pereira and Schabes, 1992; Miller and Fox, 1994; Briscoe and Carroll, 1997). However, the automatic acquisition of linguistic resources, in particular grammars, has its own problems, the most serious one being that automatically induced resources are mostly shallow and therefore of restricted use. In addition, the quality of automatically induced resources is often inferior to manually created resources. The challenge at hand consists of developing a method for automatically acquiring deep, wide-coverage linguistic resources which are able to generalise to unrestricted data and provide truly rich and deep linguistic information.

The last fifteen years have seen the development of a new and active research area working with deep grammatical frameworks like Tree Adjoining Grammar

(TAG) (Xia, 1999; Chen and Shanker, 2000), Categorial Grammar (CCG) (Hockenmaier and Steedman, 2002a), Head-Driven Phrase Structure Grammar (HPSG) (Nakanishi et al., 2004; Miyao and Tsujii, 2005) and Lexical Functional Grammar (LFG) (Cahill et al., 2002, 2003; Cahill, 2004; Cahill et al., 2005), taking up the challenge to automatically acquire deep, rich linguistic resources encoding detailed and fine-grained linguistic information from treebanks (i.e. labelled data). To date, most of the work has concentrated on English.

While the approaches mentioned above present a solution to the well-known knowledge-acquisition bottleneck by automatically inducing deep, wide-coverage linguistic resources for English, it is not clear whether the same is possible for other languages. Hockenmaier (2006) reports on the first steps on the automatic induction of rich CCG lexical resources for German. She transformed the TiGer treebank Skut et al. (1997) into a CCGbank and derived a wide-coverage CCG lexicon, but to date there are no parsing results for an automatically induced deep German CCG grammar. Burke et al. (2004b) and O'Donovan et al. (2005b) provided early and preliminary proof-of-concept research on the adaptation of the automatic LFG F-structure annotation algorithm (originally developed for English) to Spanish and Chinese, respectively. Cahill (2004); Cahill et al. (2005) ported the LFG grammar acquisition methodology to German and the TiGer treebank. The work of Cahill et al. (2003) and Cahill (2004); Cahill et al. (2005) provides proof-of-concept, showing that, in principle, the automatic acquisition of deep, wide-coverage probabilistic LFG resources for German is possible. However, the work of Cahill et al. is limited in many ways. At the time only Release 1 of the TiGer treebank was available, a preliminary, incomplete version of the treebank without morphological information. For evaluation purposes, Cahill (2004) and Cahill et al. (2003, 2005) could only revert to a hand-crafted gold standard of 100 sentences, which obviously is too small to cover many of the interesting grammar phenomena present in the full TiGer data. The most problematic aspect of their work, however, is the restricted number of grammatical features used for F-structure annotation. The set of features was rather small and coarse-grained, containing only 26 different grammatical features. Furthermore, Cahill et al. did not provide long-distance dependency (LDD) resolution for parsing.

Finally, parsing results for the automatically acquired resources for German are substantially below the results obtained for English.

This means that the question whether the automatic acquisition of truly deep, wide-coverage linguistic resources for languages different from English is possible or not, is still not fully answered. German, despite being a Germanic language and in the same language family as English, shows typological features very different from English. The main differences between the two languages concern word order and inflection: English is a configurational language with a strict Subject-Verb-Object (SVO) word order, while German shows far more flexibility with its semi-free word order. In contrast to English, which is rather poor in inflection, German morphology results in a higher number of different word forms, leading to a different distribution of word forms in the two languages, with German displaying a higher number of different word forms occurring with a low frequency only. At the same time, German has much (case) syncretism, so that despite its richer morphological inflection, German word order is in fact often highly ambiguous. These typological properties have an important impact on machine learning methods, which are the core technology in my approach for the automatic acquisition of LFG resources. It is not clear whether the methodology, which was developed for English and heavily relies on the configurational properties of English, can handle structural ambiguity and low-frequency distributions of lexical items as caused by German morphology and word order.

Besides language-specific properties, however, there is another important research challenge to treebank-based grammar acquisition. So far most of the approaches for English reported above have been based on the Penn-II treebank. This means that, to date, we do not know much about the influence of alternative treebank design, data-structures and representations, on automatic grammar acquisition. For German, Cahill (2004) and Cahill et al. (2003, 2005) based their work on the TiGer treebank (Release I), a treebank very different in design, data structures and annotation schemes from the Penn-II treebank. Here I use the TiGer treebank (Release II) as well as the TüBa-D/Z, another German treebank with newspaper text, but encoded using data structures very different from the ones in the TiGer treebank. Chapter 2 presents the two treebanks and describes the major differences between the two annotation schemes. In addition

to focussing on language-specific properties like (semi-)free word order and a rich morphological system, in my research I investigate the influence of a particular treebank annotation scheme on grammar acquisition and, in particular, on parsing, as the use of statistical parsers is a core technology in the treebank-based LFG grammar acquisition approach. In Chapter 3 I report the state-of-the-art for German data-driven CFG parsing and discuss problems specific to typological properties of German. The core questions which need to be addressed in this context are:

- Is it possible to obtain parsing results from an automatically induced German grammar in the same range as the results achieved for English? Or are there language-specific properties which make parsing of German inherently more difficult?

- What is the impact of different treebank annotation schemes on PCFG parsing? Which treebank annotation scheme is more adequate to support PCFG parsing?

Questions about the impact of language-specific properties as well as data structures and treebank encodings on data-driven parsing are a recurrent theme in my thesis. Both issues constitute open research questions and have been discussed controversely over the last years (Kübler, 2005; Maier, 2006; Kübler et al., 2006; Dubey and Keller, 2003; Schiehlen, 2004).

Recent studies by Kübler (2005); Kübler et al. (2006) and Maier (2006) investigate the influence of different treebank annotation schemes on data-driven parsing results for German and question the widely accepted assumption that lexicalisation does not support parsing of German (Dubey and Keller, 2003). The central claim of Kübler et al. is that, contrary to what has been assumed so far, given appropriate treebank data structures and encoding schemes, parsing German is not harder than parsing more configurational languages such as English. I critically review these studies in Chapter 4 and present new evidence that strongly questions the claim of Kübler et al. My approach provides a thorough evaluation of different evaluation metrics, using automatic, controlled error insertion to assess the performance of the different metrics on data structures from different treebanks.

The experiments reported in Chapter 4 show that we still do not know enough about the relationship between treebank design, particular data-driven parsing models and language-specific features. In Chapter 5 I present a thorough investigation of the two German treebanks, showing that not only the different data representations in the treebanks influence data-driven parsing and evaluation (as shown in Section 4.2), but also that the properties of the text in the two corpora as well as the differences in linguistic analysis of the same grammatical constructions, as implemented in the two annotation schemes, are crucial factors in grammar acquisition and data-driven parser evaluation. In Chapter 6 we[1] explore some of these interrelations and discuss the impact of particular design decisions on parser performance of specific grammatical constructions.

In the remaining part of my thesis I extend the research question to the adequacy of particular treebank designs for the automatic acquisition of deep, wide-coverage linguistic resources. After providing some background on treebank-based automatic acquisition of deep LFG approximations (Chapter 7), I present an improved method for treebank-based deep wide-coverage grammar acquisition for German (Chapters 8 and 9), based on and substantially revising and extending the preliminary, proof-of-concept work by Cahill et al. (2003, 2005) and Cahill (2004). I automatically extract LFG resources from two German treebanks, TiGer and TüBa-D/Z. The core question which is addressed here is:

- Which treebank design is more adequate for data-driven grammar acquisition and for the automatic acquisition of deep, wide-coverage LFG resources for German?

Parsing experiments with automatically acquired LFG grammars from the TiGer and TüBa-D/Z treebanks (Chapter 10) show that design properties of the TüBa-D/Z, like the annotation of topological fields and the encoding of non-local dependencies with the help of grammatical function labels, are not adequate to support machine learning methods as used in my grammar acquisition architecture. Results show that the flat structure of the TiGer treebank, where functional dependencies are expressed through attachment, is more suitable for automatic,

[1] Chapter 6 presents joint work with Sandra Kübler, Yannick Versley and Wolfgang Maier.

data-driven grammar acquisition. A major drawback, however, consists of the crossing branches resulting from non-local dependencies in the TiGer trees. Before extracting a PCFG, the discontiguous trees have to be converted into CFG representations. The standard technique used for conversion (Kübler, 2005) results in a lossy, shallow representation with no information about LDDs in the tree, which means that LFG resources automatically extracted based on these representations are also shallow. I compare two conversion methods to context-free representations (Chapter 11), the one of Kübler (2005) and the improved conversion method by Boyd (2007), and evaluate their impact on the grammar acquisition architecture.

In addition to the adequate representation of LDDs, there is another problem which needs to be addressed: low coverage for F-structure annotation resulting from the flat annotation in the TiGer treebank. In Chapter 11 I present a method for improving coverage based on automatically extracted subcategorisation frames. I describe the automatic extraction of subcategorisation frames (henceforth, subcat) from LFG F-structures generated from TiGer and TüBa-D/Z, following the work of O'Donovan et al. (2004, 2005a) for English, and show how these subcat frames can be used for disambiguation.

This thesis presents a method for automatically acquiring large-scale, robust, probabilistic LFG approximations for German. Chapter 12 compares the performance of our data-driven grammar acquisition architecture with the hand-crafted German ParGram LFG of Dipper (2003) and Rohrer and Forst (2006). The automatically acquired grammars substantially outperform the ParGram LFG with regard to coverage (Rohrer and Forst (2006) report 81.5% coverage on the NEGRA treebank, the automatically induced grammars achieve close to 90% coverage on the same data), but overall F-scores are higher for the hand-crafted LFG (Rohrer and Forst (2006) report upper and lower bounds in the range of 81.9-75.1% F-score on the TiGer Dependency Bank (TiGer DB), while our best TiGer DB-style grammar achieves an F-score of 72.7%). One reason for this is the low PCFG parsing results for German, especially with regard to the assignment of grammatical function labels. One component in our architecture are off-the-shelf PCFG parsers, which produce "shallow" constituency trees. The parser output is then annotated with LFG F-structure equations, resulting in deep linguistic

resources. The low parsing results for state-of-the-art parsers suggest an upper bound to the task of treebank-based grammar acquisition and LFG parsing for German.

1.1 Outline of the Thesis

The remainder of the thesis is structured as follows:

Chapter 2 gives a brief overview over the most important language-specific properties of German. It then presents the data used in this thesis: the German TiGer treebank and the TüBa-D/Z, and describes the different strategies they employ to encode the language-specific properties of German.

Chapter 3 describes the state-of-the-art in German PCFG parsing. It presents a literature review and discusses problems specific to parsing German and the strategies that have been tried to overcome these problems.

Chapter 4 presents a thorough evaluation of different evaluation metrics. I present experiments based on automatic, controlled error insertion and cross-treebank conversion, rejecting the claim (Kübler et al., 2006; Maier, 2006) that German is **not** harder to parse than English. I discuss the pitfalls of using particular evaluation measures in previous cross-treebank evaluations and show why the PARSEVAL metric Black et al. (1991), the most commonly used parser evaluation metric for constituency parsing, cannot be used for meaningful cross-treebank comparisons.

Chapter 5 concentrates on the different data structures and encoding strategies used in the TiGer and TüBa-D/Z treebanks. Having rejected the PARSEVAL metric as a valid measure for comparing treebanks with different encoding schemes, I show that other issues like out-of-domain problems and differences in linguistic analysis make a direct, automatic comparison of different treebanks infeasible.

Chapter 6 presents an extensive evaluation of three different parsers, trained on the two treebanks. An automatic dependency-based evaluation and

a human evaluation on the TePaCoC, a new testsuite for testing parser performance on complex grammatical constructions, provides new insights on the impact of flat vs. hierarchical annotation schemes on data-driven parsing.

Chapter 7 outlines previous research on treebank-based acquisition of deep LFG grammars.

Chapter 8 presents an improved method for treebank-based deep wide-coverage grammar acquisition for German, based on and substantially revising and extending the preliminary, proof-of-concept work by Cahill et al. (2003, 2005) and Cahill (2004). The chapter gives an overview of different gold standards available for German, including the DCU250, a dependency gold standard with an extended feature set for the evaluation of the LFG annotation algorithm.

Chapter 9 describes the development of an f-Structure annotation algorithm for the extended feature set in the TiGer DB, DCU250 and TUBA100 gold standards and presents results for F-structure annotation on gold treebank trees.

Chapter 10 outlines my research methodology for treebank-based LFG parsing for German. I present parsing experiments with the LFG grammars automatically acquired from the two German treebanks and discuss the impact of treebank design on grammar acquisition and parsing results for German.

Chapter 11 presents two extensions to the LFG grammar acquisition architecture: the recovery of LDDs in the parse trees and a method for improving coverage, based on subcat frames automatically extracted from LFG F-structures.

Chapter 12 discusses related work and compares the performance of the automatically extracted, treebank-based LFG grammar to a hand-crafted, wide-coverage LFG for German.

Chapter 13 concludes and outlines areas for future work.

Chapter 2

The Data

This chapter describes language-specific properties of German, two German treebanks, the TiGer treebank and the TüBa-D/Z, and the different strategies they employ to encode language-specific properties of German.

2.1 Language-Specific Properties of German

German, like English, belongs to the Germanic language family. However, despite being closely related there are a number of crucial differences between the two languages. One of them is the semi-free word order in German which contrasts with a more configurational word order in English; another (but related) difference concerns the richer morphology in German, compared to the rather impoverished English morphology. Both properties are reflected in the treebank data structures used to represent syntactic analyses of the particular languages.

In German complements and adjuncts can be ordered rather freely, while in English the assignment of predicate-argument structure is largely determined by the relative position in the sentence. While English instantiates an SVO (Subject-Verb-Object) word order, in German the position of the finite verb is dependent on the sentence type. German distinguishes three different types of sentence configuration relative to the position of the finite verb:

1. verb-first (V1, yes-no questions)

(1) War Josef gestern Nacht Salsa tanzen?
Was Josef yesterday night Salsa dancing?
Did Josef dance Salsa last night?

2. verb-second (V2, declarative clauses)

(2) Josef war gestern Nacht Salsa tanzen.
Josef was yesterday night Salsa dancing.
Josef was dancing Salsa last night.

3. verb-final (VL, subordinate clauses)

(3) Weil Josef gestern Nacht Salsa tanzen war, ...
Because Josef yesterday night Salsa dancing was, ...
Because Josef was dancing Salsa last night, ...

Non-finite verb clusters are usually positioned at the right periphery of the clause, irrespective of the sentence type. The different possibilities for verb placement increase the possibilities of parse errors.

(4) Sie begann **die Bücher** zu lesen, **die** sie gekauft hatte.
She began the books to read, which she bought had.
She began to read the books which she had bought.

Discontinuous constituents provide another difference between German and English. While both languages allow the extraposition of clausal constituents to the right periphery of a clause, this phenomenon is much more frequent in German. This is especially true for extraposed relative clauses (Example 4). Gamon et al. (2002) compare the frequency of three types of extraposed clauses in German and English technical manuals (relative clause extraposition, infinitival clause extraposition and complement clause extraposition). The most frequent phenomenon out of the three is relative clause extraposition: around one third of the relative clauses in the German manuals were extraposed, while in the English manuals extraposed relative clauses and extraposed infinitival clauses constitute less than one percent of the clause types, and extraposed complement clauses did not occur at all. Gamon et al. (2002) also report numbers for the

German NEGRA treebank (Skut et al., 1997), a German newspaper corpus. Here extraposed relative clauses account for approximately 27% of all relative clauses.

Another major difference concerns the morphological system in each language. English is poor in inflectional forms, while German shows far richer morphological variation. In contrast to English, case is marked for nouns, determiners and adjectives in German. Nominative case indicates the subject function, while the direct object is marked with accusative case. This allows for more flexibility in word order, while in English the position of the different arguments in the sentence is fixed (Table 2.1).

Der Hund$_{Nom}$ beißt den Mann$_{Acc}$.	The dog$_{Nom}$ bites the man$_{Acc}$.
Den Mann$_{Acc}$ beißt der Hund$_{Nom}$.	The dog$_{Nom}$ bites the man$_{Acc}$.
Beißt der Hund$_{Nom}$ den Mann$_{Acc}$?	Is the dog$_{Nom}$ biting the man$_{Acc}$?
Beißt den Mann$_{Acc}$ der Hund$_{Nom}$?	Is the dog$_{Nom}$ biting the man$_{Acc}$?

Table 2.1: Nominative and accusative case marking in German and English (masculine nouns)

However, morphological case is not always enough to disambiguate between different types of arguments. Consider a variation of the examples in Table 2.1 where we replace the masculine *man* (Mann) by the feminine *woman* (Frau) and the masculine *dog* (Hund) by the neutral *horse* (Pferd). In this case the surface form does not disambiguate between the subject and the direct object (this is known as case syncretism; see Table 2.2) and the sentence is ambiguous. This also increases the structural ambiguity in German.

Das Pferd$_{Nom/Acc}$ beißt die Frau$_{Nom/Acc}$.	The horse$_{Nom}$ bites the woman$_{Acc}$.
Die Frau$_{Nom/Acc}$ beißt das Pferd$_{Nom/Acc}$.	The horse$_{Nom}$ bites the woman$_{Acc}$.
Beißt das Pferd$_{Nom/Acc}$ die Frau$_{Nom/Acc}$?	Is the horse$_{Nom}$ biting the woman$_{Acc}$?
Beißt die Frau$_{Nom/Acc}$ das Pferd$_{Nom/Acc}$?	Is the horse$_{Nom}$ biting the woman$_{Acc}$?

Table 2.2: Nominative and accusative case marking in German and English (feminine and neutral nouns)

Another problem is caused by the different distribution of word forms in both languages. For German, morphological variation causes a higher number of dif-

ferent word forms which occur with low frequency in the training data. This is a problem for machine learning-based approaches and causes data sparseness for lexicalised parsing models for German (Dubey and Keller, 2003). This means that machine learning-based approaches developed for English may not generalise well to German.

2.2 Two German Treebanks: TiGer and TüBa-D/Z

The TiGer treebank (Brants et al., 2002) and the TüBa-D/Z (Telljohann et al., 2005) are two German treebanks with text from the same domain, namely text from two German daily newspapers. While TiGer contains text from the *Frankfurter Rundschau*, the TüBa-D/Z text comes from the *taz (die tageszeitung)*. The TüBa-D/Z (Release 2) consists of approximately 22 000 sentences, while TiGer (Release 2) is much larger with more than 50 000 sentences.[2] Sentence length in the two treebanks is comparable with around 17 words per sentence (Table 2.3). Both treebanks are annotated with phrase structure trees, dependency (grammatical relation) information and POS tags, using the Stuttgart Tübingen Tag Set (STTS) (Schiller et al., 1995).

	# sent.	avg. sent. length	cat. node labels	GF labels	non-term. /term. nodes
TiGer	50474	17.46	25	44	0.47
TüBa-D/Z	27125	17.60	26	40	1.20

Table 2.3: Some features of TiGer and TBa-D/Z

While both treebanks use the same POS tagset, there are considerable differences with regard to the set of syntactic categories in each treebank. TiGer has a set of 25 syntactic category labels, TüBa-D/Z distinguishes 26 different syntactic categories. The main difference between the two sets is the use of *topological*

[2]Part of the experiments reported in the thesis (Chapters 4, 6, 8) were conducted using Release 3 of the TüBa-D/Z, which was published in July 2006 and which has a size of approximately 27 000 sentences.

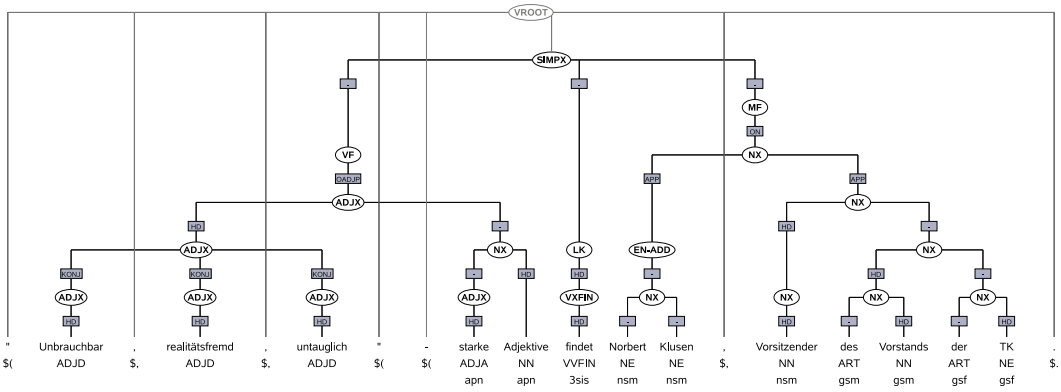
fields in TüBa-D/Z. The Topological Field Model (Herling, 1821; Erdmann, 1886; Drach, 1937; Bierwisch, 1963; Höhle, 1986) is a descriptive grammar theory, capturing the partially free German word order which accepts three possible sentence configurations (V1, V2, VL). Depending on the sentence type, the model posits the separation of a sentence into several fields (Table 2.4), where certain constraints have to be satisfied. For verb-second sentences, for example, the finite verb is positioned in the left sentence bracket (LF), while co-occurring non-finite verbs are moved to the right sentence bracket, also called the *verbal complex (VC)*. It is widely accepted that the initial field (VF) contains exactly one constituent (Berman, 1996), while there are no real restrictions for the middle field. The final field (NF) is optionally filled. For verb-last sentences the finite verb is positioned in the right sentence bracket, but this is not necessarily the last element of the sentence. Again the final field may be optionally filled. For verb-first sentences the initial field has to be empty.

	Vorfeld *initial field* **(VF)**	**Linke Satzklammer** *left sentence bracket* **(LF)**	**Mittelfeld** *middle field* **(MF)**	**Rechte Satzklammer** *right sentence bracket* **(VC)**	**Nachfeld** *final field* **(NF)**
		`Dances`	`Josef on the table`	`about?`	
V1		Tanzt	Josef auf dem Tisch	herum?	
V2	Josef	tanzt	auf dem Tisch	herum.	
V2	Josef	tanzt		herum	auf dem Tisch.
VL	weil		Josef auf dem Tisch	herumtanzt.	

Table 2.4: Topological fields and word order in German

Contrary to the basic assumptions in the Topological Field model, Müller (2005) presents data which shows that multiple frontings in German are a common phenomenon. The TüBa-D/Z annotation scheme integrates multiple constituents into one phrasal constituent and attach this constituent to the initial field (VF) (Figure 2.1).

Because of the high variability in the order of German complements and adjuncts, the syntactic annotation for both treebanks is supplemented by grammatical function labels, annotating predicate-argument structure in the trees. TiGer



"Unusable, out of touch with reality, unapt" - Norbert Klusen, chairman of the TK management, uses strong adjectives.

Figure 2.1: Multiple elements in the initial field and their annotation in TüBa-D/Z

uses 44 different grammatical labels, while TüBa-D/Z utilises 40 grammatical features (Table 2.6).

The basic arguments like subject, accusative object, prepositional objects or appositions exist in both treebanks, but they are not always used in exactly the same way. In Chapter 5.3 I will describe some of the differences in detail. The basic Topological Field Model does not support the annotation of (local or non-local) dependencies. Therefore TüBa-D/Z reverts to the use of grammatical functions to express dependency relations. This results in a set of grammatical functions with labels expressing head-dependent relationships such as *modifier of an accusative object, modifier of a modifier, conjunct of a modifier of a modifier* and so on (Table 2.6).

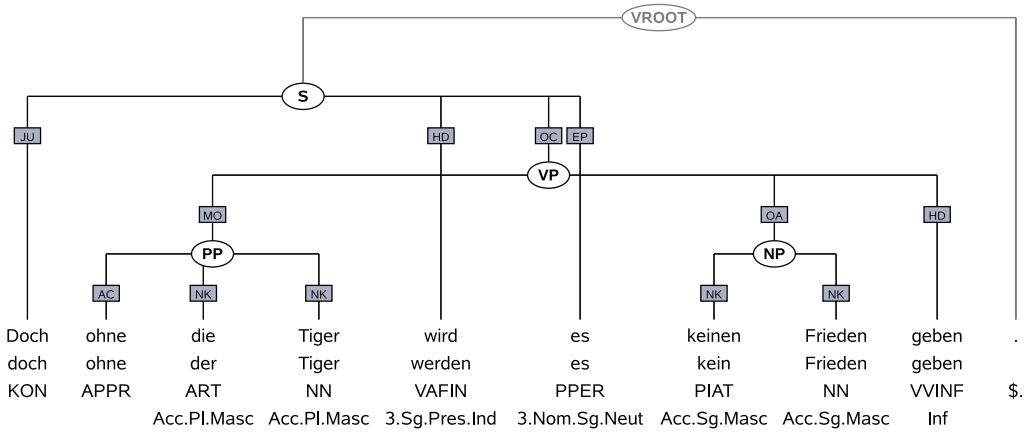
Some of the grammatical functions in the TüBa-D/Z occur with a very low frequency (OA-MODK, ON-MODK and OADVPK occur only once in 27125 sentences in TüBa-D/Z Release 3, OG-MOD 7 times, OADJP-MO 8 times, OADVP-MO 10 times, FOPPK 17 times). This poses a problem for machine learning methods, which rely on a sufficiently large set of training instances in order to achieve good performance on unseen data.

TiGer		**TüBa-D/Z**	
similar syntactic categories present in both treebanks			
AP	adjectival phrase	ADJX	
AVP	adverbial phrase	ADVX	
CH	chunk (mostly used for foreign language material)	FX	foreign language material
NP	noun phrase	NX	
PN	proper noun	EN-ADD	
PP	adpositional phrase	PX	
S	sentence	SIMPX	
VROOT	virtual root	VROOT	
topological field labels in TüBa-D/Z			
		C	field for complementiser (VL)
		FKONJ	conjunct with more than 1 field
		FKOORD	coordination of complex fields
		KOORD	field for coordinating particles
		LK	left sentence bracket
		LV	topological field for resumptive constructions
		MF	middle field
		MFE	second middlefield for substitutive infinitive
		PARORD	field for non-coordinating XX particle (V2)
coordination			
CAC	coordinated adposition	FKONJ	conjunct with more than 1 field
CAP	coordinated adjective phrase	FKOORD	coordination of complex fields
CAVP	coordinated adverbial phrase	KOORD	field for coordinating particles
CCP	coordinated complementiser		
CNP	coordinated noun phrase		
CO	coordination		
CPP	coordinated PP		
CS	coordinated sentence		
CVP	coordinated VP		
CVZ	coordinated zu-marked infinitive		
miscellaneous			
AA	superlative phrase with “am”	C	field for complementiser (VL)
DL	discourse level constituent	DM	discourse marker
ISU	idiosyncratic unit	DP	determiner phrase
MTA	multi-token adjective	P-SIMPX	paratactic coordination of 2 sent.
NM	multi-token number		
VZ	zu-marked infinitive		

Table 2.5: Syntactic category labels in TiGer and TüBa-D/Z

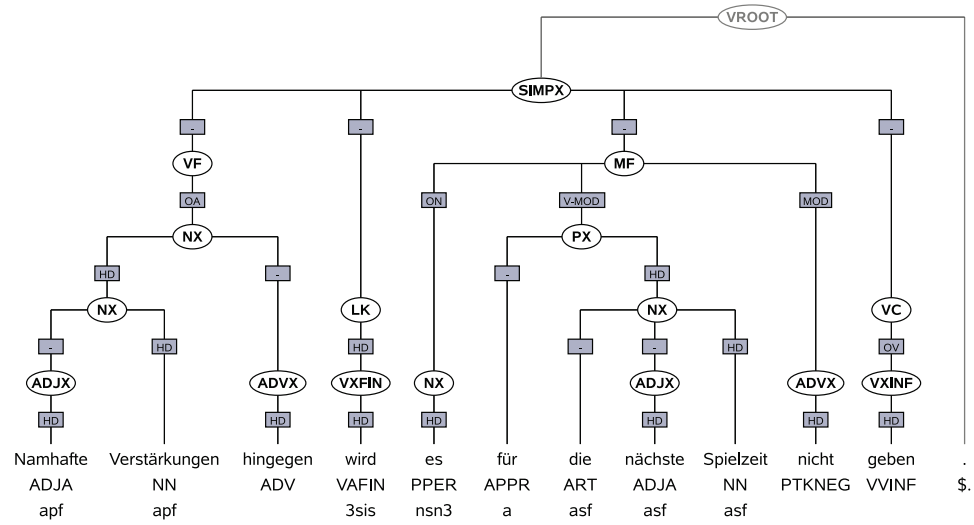
similar grammatical functions present in both treebanks			
TiGer		**TüBa-D/Z**	
SB	subject	ON	
OA	accusative object	OA	
DA	dative object	OD	
OG	genitive object	OG	
OP	prepositional object	OPP	
APP	apposition	APP	
HD	head	HD	
CJ	conjunct	KONJ	
MO	modifier	MOD	ambiguous modifier
OC	clausal object	OV	verbal object
PD	predicate	PRED	
SVP	separable verb	VPT	
grammatical functions only used in TiGer			
AC	adpositional case marker	NK	noun kernel
ADC	adjective component	NMC	numerical component
AG	genitive attribute	OA2	second accusative object
AMS	measured argument of ADJ	OC	clausal object
AVC	adverbial phrase component	PAR	parenthesis
CC	comparative complement	PG	phrasal genitive
CD	coordinating conjunction	PH	placeholder
CM	comparative conjunction	PM	morphological particle
CP	complementiser	PNC	proper noun component
CVC	collocational verb construction	RC	relative clause
DH	discourse-level head	RE	repeated element
DM	discourse marker	RS	reported speech
EP	expletive es	SBP	passivised subject (PP)
JU	junctor	SP	subject or predicate
MNR	postnominal modifier	UC	unit component
NG	negation	VO	vocative
grammatical functions only used in TüBa-D/Z			
ES	initial field-es (expletive)	OD-MOD	modifier of OD
FOPP	PP obj. (facultative)	ODK	conjunct of OD
FOPP-MOD	modifier of a FOPP	OG-MOD	modifier of OG
FOPPK	facultative obj. of FOPP	ON-MOD	modifier of ON
MOD-MOD	modifier of a MOD	ON-MODK	conjunct of ON-MOD
MODK	conjunct of MOD-MOD	ONK	conjunct of ON
OA-MOD	modifier of OA	OPP-MOD	modifier of OPP
OA-MODK	conjunct of OA-MOD	OS	sentential object
OADJP	ADJP object	OS-MOD	modifier of OS
OADJP-MO	modifier of OADJP	OV	verbal object
OADVP	ADVP object	PRED-MOD	modifier of PRED
OADVP-MO	modifier of OADVP	PREDK	conjunct of PRED
OADVPK	conjunct of OADVP-MO	V-MOD	verbal modifier
OAK	conjunct of OA	V-MODK	conjunct of V-MOD

Table 2.6: Grammatical function labels in TiGer and TüBa-D/Z



But without the Tigers will it no peace give.
"But without the Tigers there will be no peace."

Figure 2.2: TiGer treebank tree



Namable reinforcements however will it for the next playing time not give
"However, there won't be considerable reinforcements for the next playing time."

Figure 2.3: TüBa-D/Z treebank tree

Figures 2.2 and 2.3 illustrate the most important differences between the TiGer and the TüBa-D/Z annotation schemes. The constituency annotation in the TiGer treebank is rather flat and allows no unary branching, whereas the nodes in TüBa-D/Z do contain unary branches and a more hierarchical constituency structure, resulting in a much deeper tree structure than the trees in the TiGer treebank. This is reflected by the on average higher number of syntactic category nodes per sentence for the TüBa-D/Z (20.9 in TüBa-D/Z vs. 7.4 nodes per sentence in TiGer). Figures 2.2 and 2.3 show the different annotation of PPs in both annotation schemes. In TiGer (Figure 2.2) the internal structure of the PP is flat. The adjective and the noun inside the PP are directly attached to the PP, while TüBa-D/Z (Figure 2.3) is more hierarchical and inserts an additional NP node inside the PP. The PP in the TiGer sentence is a modifier of the direct object *keinen Frieden* (no peace). The relation between the two constituents is expressed through attachment: both, the PP and the accusative NP are attached to the same parent node (VP), which results in crossing branches. In the TüBa-D/Z example the PP *für die nächste Spielzeit* (for the next playing season) is a verb modifier. Due to the annotation of topological fields the two constituents end up in different fields. Here the dependency relation is expressed with the help of the complex grammatical function label V-MOD.

The differences in encoding between TiGer and TüBa-D/Z express different views on syntax: TiGer consistently encodes functor-argument structure by annotating all dependents of a head in a local tree. TüBa-D/Z, on the other hand, follows the topological field model, where the trees encode the distribution of word classes due to field constraints. As a result, predicate-argument structure is not explicitly encoded in the TüBa-D/Z trees, but can only be recovered by the help of grammatical function labels.

2.3 Differences between TiGer and NEGRA

To date, most data-driven parsing for German has been done using the NEGRA corpus as a training resource (Dubey and Keller, 2003; Fissaha et al., 2003; Schiehlen, 2004; Kübler, 2005; Versley, 2005; Maier, 2006). The annotation scheme of the TiGer treebank is based on the NEGRA annotation scheme

(Skut et al., 1997), but also employs some important extensions, which include the annotation of verb-subcategorisation, appositions and parentheses, coordinations and the encoding of proper nouns (Brants et al., 2002). The text in both corpora comes from the *Frankfurter Rundschau*, a German daily newspaper, but the NEGRA text is not a subset of the TiGer text.

2.4 Summary

In this chapter I have given an overview over the most important language-specific properties of German. I described the TiGer and TüBa-D/Z treebanks and discussed the differences in annotation schemes, resulting from the different strategies used for encoding language-specific properties of German in each of the treebanks.

The next chapter gives some background on PCFG parsing for German and reports on related work.

Chapter 3

Background and Related Work (PCFG Parsing for German)

3.1 Introduction

In early automatic parsing research, hand-crafted, symbolic, rule-based parsing approaches dominated the field (Briscoe et al., 1987; Kaplan and Maxwell III, 1988; Alshawi, 1992; Grover et al., 1993), but soon stochastic, corpus-based approaches proved to be very successful (Sampson et al., 1989; Sharman et al., 1990; Bod, 1992). The English Penn-II treebank (Marcus et al., 1993) substantially advanced the development of data-driven parsing (Magerman, 1995; Charniak, 1996; Collins, 1997). Parser F-scores, measured within the PARSEVAL metric (Black et al., 1991), have increased from around 85% (Magerman, 1995) up to more than 90% (Charniak et al., 2006; Petrov and Klein, 2007). A variety of research questions have been addressed, including the impact of lexicalisation on parsing results (Hindle and Rooth, 1993; Collins, 1997; Klein and Manning, 2003), and the role of domain variation (Gildea, 2001; Roark and Bacchiani, 2003; Judge et al., 2005; Versley, 2005). Recently, further improvements have been achieved by applying reranking techniques (Charniak and Johnson, 2005), self-training (Bacchiani et al., 2006), or combinations of both (McClosky et al., 2006a,b; Foster et al., 2007), especially to overcome out-of-domain problems.

To date, most of the parsing research has been using Penn-II treebank Wall Street Journal data. The predominance of Penn-II data lead some (Oepen, 2007)

to claim that research on statistical parsing has degenerated to the science of the Wall Street Journal, focussing on outdated, highly domain-specific text with linguistically insufficient annotation, and that this kind of research is incapable of providing us with interesting insights into human language processing, or with generalisations to other markedly different languages.

Another major source of criticism addresses parser evaluation. The standard evaluation metric for assessing constituency-based parser performance is the PARSEVAL metric (Black et al., 1991). PARSEVAL counts matching brackets in the original treebank trees and the parser output. Results report precision, recall and the number of crossing brackets in the parser output. PARSEVAL has often been criticised for not reflecting a linguistically motivated view of parser output quality. For example, it is not completely clear to what extent an improvement of 2% PARSEVAL F-score reflects an increase in quality in parser output. Another point of criticism is PARSEVAL's inability to distinguish between linguistically more or less severe errors. Carroll and Briscoe (1996) point out that PARSEVAL is very indulgent towards parser errors concerning the misidentification of arguments and adjuncts, but at the same time severely punishes rather harmless attachment errors if they are embedded deep in the parse tree. It is becoming increasingly clear that, instead of giving a linguistically motivated account of parser output quality, the PARSEVAL metric is highly sensitive to the data structures and encoding of the input data. Several proposals have been made to overcome the shortcomings of PARSEVAL (Lin, 1995; Carroll et al., 1998; Lin, 1998; Sampson and Babarczy, 2003), some driven by the conviction that not only the PARSEVAL metric, but constituency-based evaluation in general is problematic and not the road to success for a meaningful evaluation of parser output. Despite such efforts, PARSEVAL remains the standard evaluation measure for constituency-based parsing.

3.2 State-of-the-Art for German Data-Driven Parsing

The question as to less-configurational languages like German are harder to parse than English is a long-standing and unresolved issue in the parsing literature. Several studies have addressed this topic and have arrived at quite controversial conclusions (Dubey and Keller, 2003; Fissaha et al., 2003; Cahill, 2004; Abhishek and Keller, 2005). Most of the work on data-driven parsing for German to date has used the NEGRA treebank (Skut et al., 1997), a predecessor of the TIGER treebank, which is characterised by its flat tree structure and the rich set of grammatical functions.

Different strategies have been applied to the task of parsing German, some of them more successful than others. Some studies (Cahill et al., 2003; Fissaha et al., 2003; Cahill, 2004; Schiehlen, 2004; Versley, 2005) have tried to include grammatical functions or morphology in their parsing systems. Others have explored lexicalised parsing models (Dubey and Keller, 2003; Abhishek and Keller, 2005) or used treebank transformation techniques such as parent-encoding, Markovisation or split & merge operations on trees (Petrov and Klein, 2007, 2008; Rafferty and Manning, 2008).

One of the first treebank-based parsing experiments on German was conducted by Fissaha et al. (2003). They addressed the differences between NEGRA and the Penn-II treebank, namely the flat annotation which captures the partially free word order in German and the richer set of grammatical functions in the NEGRA treebank. In their experiments they explored the impact of grammatical functions on parsing results. Furthermore, they presented treebank transformations using a partial parent encoding technique, following Johnson (1998). Fissaha et al. (2003) trained the LoPar parser (Schmid, 2000) on the NEGRA treebank, using an unlexicalised probabilistic parsing model with gold POS tags as parser input. Their results showed that including grammatical functions in the training data improved parsing results in the range of 2% labelled F-measure,[3] compared to a parser trained on a grammar with syntactic categories only. Results for three

[3]The evaluation has been performed using `evalb` (Sekine and Collins, 1997), an implementation of the PARSEVAL metric.

different types of parent-encoding also improved precision, but at the cost of a dramatic decrease in coverage. Most interestingly, the authors could not detect any learning effect for their parent encoding experiments.

Fissaha et al. (2003) also addressed the question whether German is harder to parse than English. They compared their parsing results (labelled precision and recall) to state-of-the-art parsing results for a parser trained on the English Penn-II treebank, which are considerably higher. The authors put the differences down to the different treebank sizes and, perhaps optimistically, expect that the differences in performance will be reduced when training on a larger data set.

A somewhat less optimistic conclusion is reached by Dubey and Keller (2003), who discussed the role of lexicalisation for parsing models for German. They showed that, contrary to English and some other languages, lexicalisation does not improve data-driven parsing for German. In their experiments with lexicalised probabilistic grammars, Dubey & Keller were not able to outperform the baseline result obtained with an unlexicalised PCFG on the same data. They also showed that this was not due to a lack of training data. The authors suggested that the effect is caused by the flat annotation in NEGRA, which cannot be captured well by the lexicalised models which have been developed for the more hierarchical annotation scheme of the Penn-II treebank. To tackle the problem they proposed an alternative to Collins's head-head relationships, based on the treatment of non-recursive NPs in Collins (1997). Their model, called *sister-head dependencies*, implicitly adds binary branching to the flat rules in NEGRA by conditioning the probability of a rule not on the head sister but on features of the previous sister node. The *sister-head dependencies* model outperforms the unlexicalised baseline and achieves an F-score of up to 74%.

Dubey and Keller (2003) also noted that the higher parsing results achieved for the Penn-II treebank might reflect the properties of the annotation schemes. The Penn-II treebank contains hierarchical PPs, which in contrast to the flat PP annotation in NEGRA, are easier for the parser to process. Therefore Dubey and Keller (2003) claimed that parsing results for parsers trained on annotation schemes as different as NEGRA and the Penn-II treebank do not allow for a direct comparison.

Based on the observation that lexicalisation does not support data-driven parsing for German (Dubey and Keller, 2003), Schiehlen (2004) presents parsing models based on different treebank transformations to boost parser performance. His work is inspired by Klein and Manning (2003), who showed that unlexicalised parsing for English can yield results close to state-of-the-art lexicalised parsing models for English, when applying linguistically motivated splits to the treebank in order to weaken the independence assumption of PCFGs and to encode local context information in the trees. In addition to annotation strategies, Schiehlen also applies treebank transformation techniques like parent and grandparent encoding (Johnson, 1998) and Markovisation. He optimises his grammars with respect to a dependency-based evaluation and shows that constituency-based and dependency-based evaluation results do not always agree. He also shows that, while improving scores for constituency-based evaluation, parent-annotation and Markovisation do impair results for word-word dependencies in the parser output. Schiehlen explains this by the flat annotation in the NEGRA treebank, which does not gain much from parent-encoding techniques. In fact, transformations worsen the problem of sparse data which, due to the high number of long low-frequency rules, is already an issue for the NEGRA treebank. Markovisation, on the other hand, takes away necessary context information from the trees. In German, in contrast to English, predicate-argument structure can not be determined locally. Therefore Schiehlen claims that Markovisation, despite working for the English Penn-II treebank, does not work for a parser trained on the NEGRA treebank.

Kübler et al. (2006) return to the question of lexicalisation and challenge the claim that lexicalised parsing does not work for German. They present experiments contradicting Dubey and Keller (2003), showing that lexicalisation does support data-driven parsing for German when using the Stanford parser (Klein and Manning, 2003), a state-of-the-art probabilistic parser which provides a factored probabilistic model combining a PCFG with a dependency model. They trained the parser on NEGRA and on the TüBa-D/Z. For both treebanks they obtained a slight improvement for the lexicalised parsing model. However, the improvement for the NEGRA treebank was only in the range of 0.2 labelled F-score, which is unlikely to be statistically significant. For the more hierarchical TüBa-D/Z the improvement was more profound at 2.4%. But, considering that

the Stanford parser offers a number of features like vertical and horizontal Markovisation,[4] it is not clear whether the improvement can in fact be traced back to the use of lexical information only.

Kübler et al. (2006) present further parsing experiments with three different parsing models (Stanford unlexicalised, Stanford lexicalised, LoPar unlexicalised) and show that `evalb` F-scores for all models for the parsers trained on NEGRA are between 15 and 20% lower compared to the parsers trained on the TüBa-D/Z, which obtain parsing results in the same range as parsers trained on the Penn-II treebank. Kübler et al. (2006) conclude that German is not harder to parse than English and that low parsing results for the NEGRA treebank are an artefact of encoding schemes and data structures rather than due to language-specific properties. I will come back to this topic in Chapter 4, showing why the claim by Kübler et al. (2006) does not hold.

Petrov and Klein (2008) achieve the best PARSEVAL scores for both German treebanks, TiGer and TüBa-D/Z, in a shared task on Parsing German (PaGe) (Kübler, 2008). They use a latent variable method, a language-agnostic approach based on automatically refining and re-annotating the original treebank by a number of split & merge operations, so that the likelihood of the transformed treebank is maximised. Petrov and Klein (2008) compare two different approaches for assigning grammatical functions. In the first approach they merge the grammatical function labels with the syntactic node labels, resulting in new, atomic node labels. In the second approach they first train their parser on a version of the treebank which has been stripped of grammatical functions. After 4 training iterations which apply the split & merge technique, their grammars achieve good accuracy on constituent labels. In a second pass they assign grammatical functions to the constituent trees. Most interestingly, the two-pass parsing approach yields much lower results than the ones for the merged node–grammatical function labels. Petrov & Klein explain this by the fact that grammatical functions model long-distance dependencies, while the two-pass model, which uses split &

[4] Horizontal Markovisation (Schiehlen, 2004) decomposes the grammar rules by constraining the horizontal context of each rule. Vertical Markovisation (also called parent-annotation (Johnson, 1998)), on the other hand, adds vertical context to the rules by adding the syntactic category of the parent node to each node in the tree.

merge operations during the first pass only and assigns the grammatical functions using a local X-Bar style grammar, is not good at capturing non-local relations.

3.2.1 Morphological Information

An approach which has not been tried for English (for obvious reasons) is the enrichment of the parsing models with morphological information. Cahill (2004), Schiehlen (2004) and Versley (2005) present a somewhat simplistic way of integrating morphological information into the syntactic node labels of their grammars and report contradicting results.

As Cahill (2004) and Schiehlen (2004) both work with a treebank which does not include explicit morphological annotation (NEGRA and TiGer Release 1, respectively), they automatically simulate morphological information in the trees. They exploit functional annotations in the treebanks and percolate case information, which is implicitly encoded in the grammatical function labels, down to the leaf nodes. Cahill (2004) annotates POS tags like determiners, adjectives and pronouns with case information, while Schiehlen (2004) assigns case marking to the categorial nodes themselves and, for NPs, also to NP-internal common nouns and pronouns. Grammatical function labels triggering such a transformation are SB, PD and SP (nominative), OA and OA2 (accusative), DA (dative), and AG and OG (genitive).

Cahill (2004) did not observe any improvement over parsing models without case information. She puts this down to the incompleteness and coarseness of the grammar transformation and expects better results for a more detailed and complete morphological analysis. In contrast to Cahill (2004), the results of Schiehlen (2004) show a clear improvement of around 4% for a constituency-based evaluation and around 3% for a dependency-based evaluation. It is not clear whether the contradictory results are due to the differences with respect to the tree transformations, the different sizes of the training sets (Cahill trained on a TIGER training set of about twice the size of the NEGRA treebank) or the parsing models themselves (Schiehlen's PCFG includes grammatical function labels only for the case-marking transformations described above, while Cahill uses an LFG f-structure-annotated PCFG with far more information; Cahill's

model integrates grammatical functions and LFG f-structure annotations into the syntactic node labels).

Cahill (2004) and Schiehlen (2004) try to improve parser accuracy for German by enriching the node labels with case information. Dubey (2005) presents a different approach to include morphology into the parsing model. He provides a special treatment for unknown words by the means of a suffix analyser (Brants, 2000). Results show that the suffix analysis does improve parser performance, but only after applying a number of linguistically motivated treebank transformation strategies. In contrast to Schiehlen (2004), who argued that Markovisation does not help for the German NEGRA treebank, Dubey (2005) achieves better results for a Markovised grammar induced from NEGRA. However, Dubey (2005) presents a constituency-based evaluation only, so the question whether Markovisation does help for parsing German in general (i.e. also for a dependency-based evaluation) cannot be answered here. Versley (2005) addresses this issue by presenting parsing experiments for German across different text types. Like Schiehlen (2004) and Dubey (2005), he applies a number of linguistically motivated treebank transformations. In his experiments Markovisation gives a slight improvement for the transformed grammar (dependency evaluation), while it hurts performance for a vanilla PCFG. Case marking, included in the syntactic node labels of NPs as well as the POS tag labels of determiners and pronouns, also helps for all different text types.

So far the literature on parsing German has reported a rather confusing picture of the usefulness of different features like grammatical functions, lexicalisation, Markovisation, split & merge operations and morphology for boosting parsing performance for German. Rafferty and Manning (2008) follow up on this and try to establish baselines for unlexicalised and lexicalised parsing of German, using the Stanford parser (Klein and Manning, 2003) with different parameter settings, trained on the German TiGer and TüBa-D/Z treebanks. The results obtained, however, do not settle the case but rather add to the confusion. What becomes clear is that the three settings tested in the experiments (Markovisation, lexicalisation and state splitting) strongly interact with each other, and also with a number of other factors like the size of the training set, the encoding and, in particular, the number of different categorial node labels to be learned by the

parser. This number crucially increases when including grammatical function labels in the categorial node labels. It becomes apparent that especially the TiGer treebank suffers from a sparse data problem, caused by the flat trees, and that smoothing could present a possible way out of the dilemma. This is consistent with Dubey (2004, 2005), who achieves considerable improvements by experimenting with different smoothing techniques.

Rafferty and Manning (2008) present no dependency-based evaluation but PARSEVAL F-scores only, which leads them to conclude that including grammatical functions in the parsing model increases data sparseness and therefore reduces parser performance by 10-15%. The inclusion of grammatical functions into the node labels results in a set of 192 (instead of 24) syntactic category labels for TiGer, which have to be learned by the parser. Therefore, a decrease in F-score is not surprising. However, due to the variability of the relatively free order of complements and adjuncts in German, it is not sufficient to identify say an NP node label with the correct phrase span. In order to recover the meaning of a sentence, it is also necessary to distinguish arguments from adjuncts, and to identify the grammatical function of each argument. Therefore it is arguable whether higher F-scores for an impoverished parser output present useful information, or whether lower scores for a more meaningful representation are, in fact, better.

3.2.2 The Pitfalls of Evaluation

The considerations above raise the question of what are valid methods for the evaluation of different parsing models, particularly so for cross-treebank and cross-language comparisons involving different annotation schemes. Based on the observation that the constituency-based PARSEVAL F-measure does not necessarily correspond to an improvement for a dependency evaluation (Schiehlen, 2004), I consider pure constituency-based EVALB F-scores insufficient to compare different parsing systems. There are some well-known drawbacks, like for example the tendency of PARSEVAL towards errors concerning the identification of complements and adjuncts (Carroll and Briscoe, 1996), or that PARSEVAL shows a varying tolerance towards attachment errors, depending on how deep they are embedded

within the tree (Manning and Schütze, 1999). It is also not always clear how to interpret PARSEVAL F-scores. Intuition tells us that higher scores reflect higher quality in the parser output, but it is by no means evident that this always holds and, if so, to what extent, as there is not yet a proven correlation between human judgements on parser output quality and PARSEVAL F-scores. The F-measure often tempts us to compare apples with oranges: Fissaha et al. (2003) and Kübler et al. (2006) for example compare F-scores for the English Penn-II treebank and the German NEGRA treebank. I will return to this issue in Chapter 4. Proposals have been made to overcome the weaknesses of the PARSEVAL metric, see for example (Lin, 1995, 1998; Carroll et al., 1998; Kübler and Telljohann, 2002; Sampson et al., 1989; Sampson and Babarczy, 2003). I will provide a detailed discussion of evaluation alternatives in Chapter 4.

Coming back to the topic of state-of-the-art parsing systems for German, it is not straightforward to decide which system provides the best data-driven parsing results for German. Petrov and Klein (2008) achieve best PARSEVAL scores in a shared task (Kübler, 2008) with a language independent latent variable method.

It might seem ironic that a language-independent approach scores best for the task of parsing German. However, this is not as strange as it seems: the method does not rely on any predefined linguistic knowledge but uses a split-and-merge technique which automatically refines the treebank and finds the optimal encoding for each particular treebank annotation scheme. So the grammar extracted for German would have different properties compared to the one extracted for the English Penn-II treebank, as would each grammar induced from the different treebanks available for German.

3.2.3 Significance Tests for Parser Performance

Another issue for parser evaluation is the question of how to decide whether an increase or decrease in parser output results is statistically significant or not. Dan Bikel provides software[5] working on EVALB output for two different parsing runs, which outputs p-values for whether observed differences in recall and/or precision are statistically significant. The program uses a *compute-intensive randomised*

[5]Available at: `http://www.cis.upenn.edu/~dbikel/software.html`.

test, in which the null hypotheses (the two models that produced the observed results are the same) is tested by randomly shuffling scores for individual sentences between the two models and then re-computing precision and recall for the new result sets. For each shuffle iteration, a counter is incremented if the difference in results after shuffling is equal to or greater than the original observed difference. After 10,000 iterations, the likelihood of incorrectly rejecting the null hypothesis is computed as follows:

$$(nc + 1)/(nt + 1), \tag{3.1}$$

nc is the number of random differences greater than the original observed difference, and nt is the total number of iterations.

In my thesis, however, I did not perform any significance tests for the results of my parsing experiments. I argue that the PARSEVAL metric does not provide a meaningful evaluation of parser output quality for cross-treebank comparisons (see Chapter 4), therefore it seems somehow pointless to perform significance tests for results which, in itself, are not meaningful.

For evaluating parsers trained on the same treebank, significance tests seem to be more informative. This, however, is not necessarily true. Let us assume that we have two different parsers which have been trained on the same data, thus parsing raw text into the same type of tree representations, using the same set of syntactic categories. We use these parsers to obtain a syntactic analysis for the sentence in Example (5).

(5) So erklärt Edward Brandon vom Unternehmen National City:
so explains Edward Brandon of the company National City:
Edward Brandon of National City thus explains:

Let us further assume that the first parser has access to an external resource for Named Entity Recognition, thus correctly annotating *Edward Brandon* and *National City* as proper nouns (PN) (Example 6), while the second parser analyses the same constituents as noun phrases (NP) (Example 7). The second parser, on the other hand, might have a more sophisticated way to deal with PP attachment, and so correctly attaches the PP *vom Unternehmen* to the noun *Brandon*, but

fails to identify *Edward Brandon* and *National City* as named entities but projects each of the two contituents to an NP node (Example 7).

(6) (TOP (S (ADV So) (VVFIN erklrt) (NP (PN (NE Edward) (NE Brandon))) (PP (APPRART vom) (NN Unternehmen) (PN (NE National) (NE City)))) (PUNC :))

(7) (TOP (S (ADV So) (VVFIN erklrt) (NP (NN Edward) (NN Brandon) (PP (APPRART vom) (NN Unternehmen) (NP (NN National) (NN City))))) (PUNC :))

From a linguistic point of view, we would prefer the analysis in (7), where PP attachment has been analysed correctly, while the difference between an NP and a proper name node is not as crucial for understanding the meaning of the sentence. PARSEVAL, however, would evaluate the two parses as follows (Table 3.1), giving better results to the analyis in 6:

	Precision	**Recall**	**F-score**
(6)	83.3	83.3	83.3
(7)	80.0	66.7	72.7

Table 3.1: PARSEVAL results for Examples 6 and 7

It has yet to be shown whether PARSEVAL provides a meaningful evaluation of parser output quality even for parsers trained on the same treebank. Therefore I do not test for statistical significance of parsing results in my experiments, as these results might be misleading.

3.3 Conclusions

In this chapter I gave an overview of state-of-the-art data-driven parsing for German and discussed the different approaches used to tackle language-specific characteristics as well as treebank-specific properties. While considerable progress has been made during the last couple of years, there is still no agreement as to the impact of different strategies like lexicalisation or Markovisation on parsing German. Two major points are apparent: (1) Linguistically motivated annotation

strategies can boost parser performance to some extent. This is best done by letting the parser learn its own optimisation strategies. (2) There is a complex interaction between the different strategies to improve parsing results discussed in this section. It is not straightforward to decide whether a particular approach is useful or not. While it might be useful in a certain context, after changing some of the experimental settings, the same strategy might hurt results. Easy answers are not at hand.

In the next chapter I will focus on the question whether German is harder to parse than English or not. I provide an extensive evaluation of different evaluation metrics, based on experiments with automatic, controlled error insertion and cross-treebank conversion. I discuss the pitfalls of using particular evaluation measures in previous cross-treebank evaluations. My experiments show why the PARSEVAL metric cannot be used for meaningful cross-treebank comparisons.

Chapter 4

Evaluating Evaluation Measures

4.1 Introduction

A long-standing and unresolved issue in the parsing literature is whether parsing less-configurational languages is harder than (say) parsing English. German is a case in point. Results from Dubey and Keller (2003) suggest that, in contrast to English and other languages like French (Abhishek and Keller, 2005), (head-)lexicalisation (Dubey and Keller, 2003) does not boost performance for German parsing models. Recent results from Kübler et al. (2006) question this claim, raising the possibility that the gap between the PARSEVAL results for TiGer and TüBa-D/Z might be an artefact of encoding schemes and data structures of the treebanks which serve as training resources for probabilistic parsers. Kübler (2005); Kübler et al. (2006) and Maier (2006) show that treebank annotation schemes have a considerable influence on parsing results. A comparison of unlexicalised PCFGs trained and evaluated on the German NEGRA and the TüBa-D/Z treebanks using the LoPar parser (Schmid, 2000) shows a difference in parsing results of about 16% for a constituency-based evaluation with the PARSEVAL metric (Black et al., 1991). Kübler et al. (2006) and Maier (2006) conclude that, contrary to what had been assumed, German is not actually harder to parse than English, but that the NEGRA annotation scheme does not support optimal PCFG parsing performance.

This claim is based on the assumption that PARSEVAL is a valid measure for cross-treebank evaluation. This chapter, by using a novel approach measur-

ing the effect of controlled error insertion on treebank trees and parser output from different treebanks, shows that this claim does not hold. The error insertion approach allows for a meaningful comparison of the performance of different evaluation metrics on the different treebanks.

In the first section of this chapter I present a number of parsing experiments with controlled error insertion using the PARSEVAL metric, the Leaf-Ancestor metric as well as a dependency-based evaluation. I also provide extensive cross-treebank conversion, crucially operating on parser output, rather then on training resources, as in previous research. The results of the experiments show that, contrary to Kübler et al. (2006) the question whether or not German is harder to parse than English is still undecided.

Part of the research presented in this Chapter has been published in Rehbein and van Genabith (2007a) and Rehbein and van Genabith (2007c).

4.2 Controlled Error Insertion Experiments for German

In the parsing community, implementations of the PARSEVAL metric (Black et al., 1991) constitute the *de facto* standard constituency evaluation metric for data-driven parser performance. Despite being the standard metric, PARSEVAL has been criticised for not representing "real" parser quality (Carroll and Briscoe, 1996; Sampson, 2000; Sampson and Babarczy, 2003). The PARSEVAL metric checks label and wordspan identity in parser output compared to the original treebank trees. It neither weights results, differentiating between linguistically more or less severe errors, nor does it give credit to constituents where the syntactic categories have been recognised correctly but the phrase boundary is slightly wrong.

With this in mind, I question the claim (Kübler, 2005; Kübler et al., 2006; Maier, 2006) that the PARSEVAL results for NEGRA and TüBa-D/Z reflect a real difference in quality between the parser output for parsers trained on the two different treebanks. As a consequence I also question the claim that PARSEVAL results for German in the same range as the parsing results for the English

Penn-II treebank prove that German is not harder to parse than the more configurational English. To investigate this issue I present three experiments on the German TiGer and the TüBa-D/Z treebanks. In the first experiment I automatically insert controlled errors into the original treebank trees from TiGer and TüBa-D/Z and evaluate the modified trees against the gold treebank trees. Experiment II presents cross-treebank conversion of the parser output of a statistical parser trained on the two treebanks, and in the third experiment I supplement the previous constituency-based evaluation with PARSEVAL and LA by a dependency-based evaluation of the parser output.

4.3 Experiment I

Experiment I is designed to assess the impact of identical errors on the different encoding schemes of the TiGer and TüBa-D/Z treebanks and on the PARSEVAL and Leaf-Ancestor evaluation metrics.

4.3.1 Experimental Setup

The TiGer treebank and the TüBa-D/Z both contain newspaper text, but from different German newspapers. To support a meaningful comparison we have to compare similar sentences from both treebanks. Similarity can be understood with regard to different aspects of likeness: vocabulary, text genre, topics, syntactic structure, style, and so on. We are interested in the impact of encoding schemes on parsing results and thus define similarity with respect to the underlying syntactic structure of the sentences. Therefore I created "comparable" test sets as follows.

First I selected all sentences of length $10 \leq n \leq 40$ from both treebanks. For all sentences I extracted the sequence of POS tags underlying each sentence. Then I computed the Levenshtein edit distance (Levenshtein, 1966), a string-based similarity measure, for all lists of part-of-speech tags with equal length from the two treebanks.[6]

[6]The Levenshtein distance was computed with the help of Josh Goldberg's perl module *Text-LevenshteinXS-0.03* (`http://search.cpan.org/~jgoldberg/Text-LevenshteinXS-0.03`)

Symbol	STTS POS tags
a	ADJA ADJD
b	ADV PAV PWAV
c	APPR APPRART APPO APZR
d	ART CARD
e	ITJ
f	KOUI
g	KOUS
h	KON
i	KOKOM
j	NN NE FM TRUNC
k	PDAT PIAT PIDAT PWAT
l	PDS PIS PPER PWS
m	PPOSS
n	PPOSAT
o	PRELS
p	PRELAT
q	PRF
r	PTKZU
s	PTKNEG
t	PTKVZ
u	PTKANT
v	PTKA
w	VVFIN VMFIN VAFIN
x	VVIMP VAIMP
y	VVINF VMINF VAINF
z	VVIZU
ä	VVPP VAPP VMPP
ö	XY
ü	$. $($,

Table 4.1: Generalisations over POS tags used for conversion

The Levenshtein edit distance compares two strings (or any two lists of atomic expressions) by calculating the number of substitutions, deletions or insertions ("edits") needed to transform one string into another string. Identical strings

have an edit distance of 0. The Levenshtein distance works on strings, so the sequence of POS tags had to be converted into a sequence of one-symbol-per-POS. To avoid a sparse-data problem I applied a generalisation over POS tags: all punctuation marks were converted into the same symbol, the same was done with attributive and predicative adjectives, and so on (see Table 4.1 for a complete list of conversions).

I approximated the distribution of sentence length in both treebanks by, for each sentence length n with $10 <= n <= 40$, taking the average number of sentences with length n between the two treebanks, normalised by corpus size. Then I chose the sentences with the lowest edit distance for each particular sentence length. This resulted in two test sets with 1000 sentences each, comparable with regard to sentence length, syntactic structure and complexity distribution. Next I automatically inserted different types of controlled errors into the original treebank trees in the test sets and evaluated the modified trees against the original treebank trees, which allowed me to assess the impact of similar (controlled for type and number) errors on the two treebank encoding schemes. Grammatical function labels were not included in the evaluation.

4.3.2 Error Insertion

The inserted errors fall into three types: attachment, span and labelling (Table 4.2). The attachment errors and span errors are linguistically motivated errors which partly represent real ambiguity in the data and are also typical parser errors. Label errors are not as frequent in the parser output, but allow us to insert a high number of the same error type in both test sets and so to quantify the impact of similar errors on the results of our evaluation. The same number of errors were inserted in both test sets.

I inserted two different types of PP attachment errors: for the first type (ATTACH I) I attached all PPs which were inside of an NP one level higher up in the tree (this usually means that noun attachment is changed into verb attachment, see Figure 4.1); for the second type (ATTACH II) I selected PPs which directly followed a noun and were attached to an S or VP node (TiGer) or to the middle field (TüBa-D/Z) and attached them inside the NP node governing

	Error description
ATTACH I	Attach PPs inside an NP one level higher up in the tree
ATTACH II	Change verb attachment to noun attachment for PPs on sentence level, inside a VP or in the MF (middle field)
LABEL I	Change labels of PPs to NP
LABEL II	Change labels of VPs to PP
LABEL III	Change labels of PNs to NP
SPAN I	Include adverb to the left of a PP into the PP
SPAN II	Include NN to the left of a PP into the PP

Table 4.2: Error description for inserted error types

the preceeding noun. This usually resulted in a change from verb attachment to noun attachment (Figure 4.2).

The three types of label errors simply change the labels of PP nodes to NP (LABEL I), of VPs to PP (LABEL II) and of proper name nodes (PN) to NP (LABEL III). For the last error type I slightly changed the phrase boundaries in the trees. For SPAN I, I selected adverbs which were positioned at the left phrase boundary of a PP and included them into the PP. For SPAN II-type errors I did the same with nouns, including them in a prepositional phrase positioned to the right of the noun.

4.3.3 Results for Controlled Error Insertion for the Original Treebank Trees

Table 4.3 shows the number of errors generated and the impact of the error insertion into the original treebank trees on PARSEVAL results, evaluated against the gold trees without errors. PARSEVAL results in all experiments report labelled F-scores based on precision and recall. The first error type (PP attach-

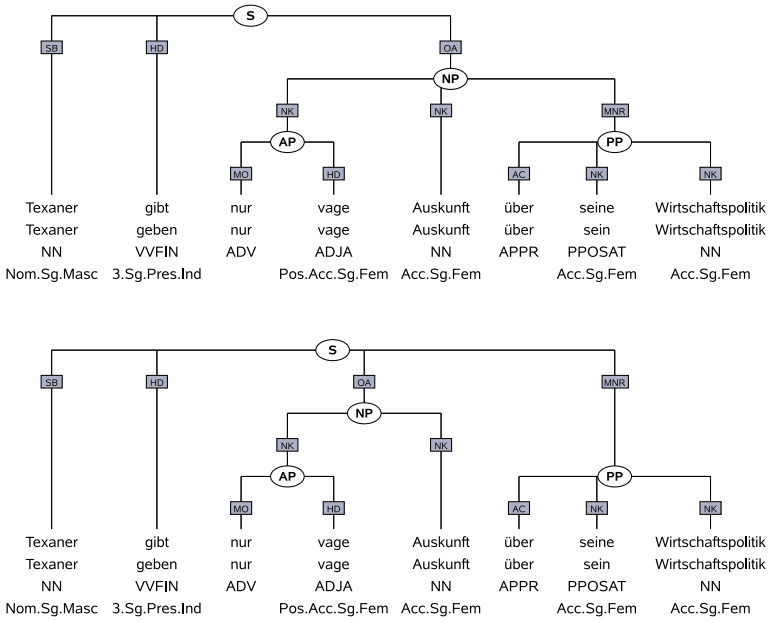


Figure 4.1: ATTACH I: changing PP noun attachment to verb attachment (TiGer example)

ment I, 593 inserted errors) leads to a decrease in F-score of 2.5 for the TiGer test set, while for the TüBa-D/Z test set the same error causes a decrease of 0.8 only. The effect remains the same for all error types and is most pronounced for the category label errors, because the frequency of the labels resulted in a large number of substitutions. The total weighted average over all error types shows a decrease in F-score of more than 18% for TiGer and of less than 8% for TüBa-D/Z. This clearly shows that the PARSEVAL measure punishes the TiGer treebank annotation scheme to a greater extent, while the same number and type of errors in the TüBa-D/Z annotation scheme do not have an equally strong effect on PARSEVAL results for similar sentences.

Experiment I shows that the gap between the PARSEVAL results for the two annotation schemes does not necessarily reflect a difference in quality between the trees. Both test sets contain the same number of sentences with the same sentence lengths. The sentences are equivalent with regard to complexity and

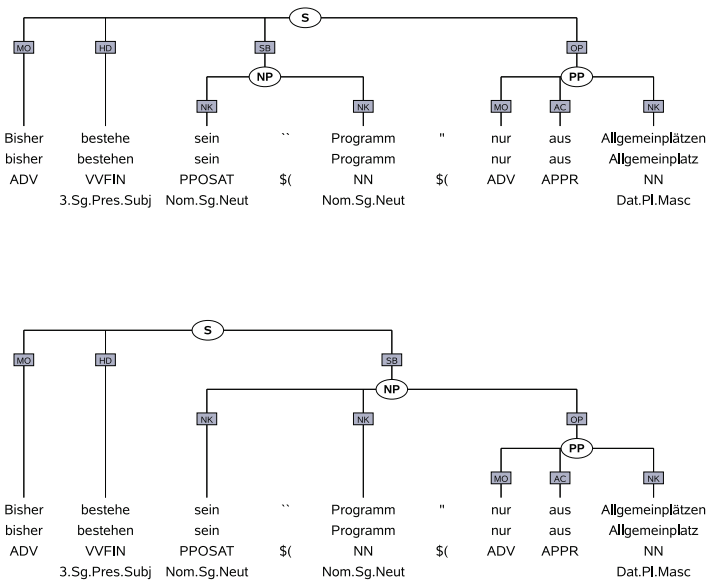


Figure 4.2: ATTACH II: changing PP verb attachment to noun attachment (TiGer example)

structure, and contain the same number and type of errors. This suggests that the difference between the results for the TiGer and the TüBa-D/Z test set is due to the higher ratio of non-terminal/terminal nodes in the TüBa-D/Z trees reported in Table 2.3.

4.3.4 The Leaf-Ancestor Metric (LA)

In order to obtain an alternative view on the quality of the annotation schemes I used the leaf-ancestor (LA) metric (Sampson and Babarczy, 2003), a parser evaluation metric which measures the similarity between the path from each terminal node in the parse tree to the root node and the corresponding path in the gold tree. The path consists of the sequence of node labels between the terminal node and the root node, and the similarity of two paths is calculated with the help of the Levenshtein edit distance (Levenshtein, 1966).

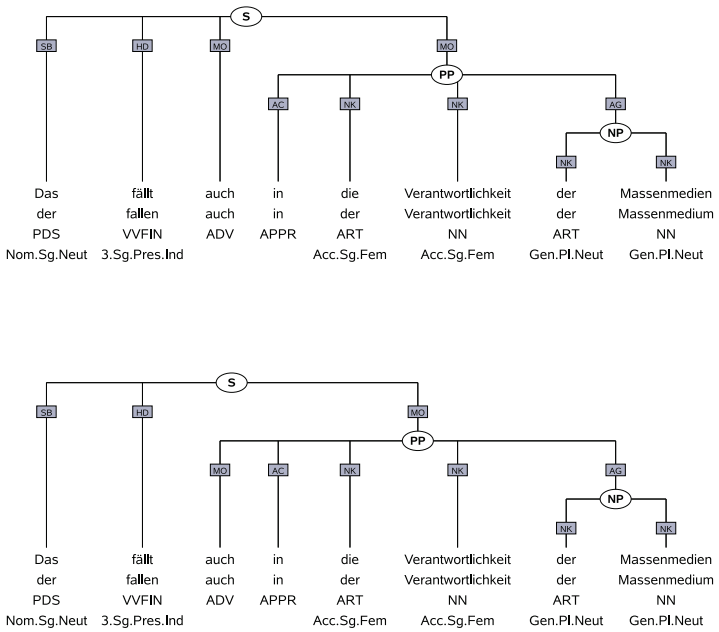


Figure 4.3: SPAN I: changing phrase boundaries (TiGer example)

Consider the following two example sentences (Figure 4.4). Let us assume that the first sentence was taken from the gold standard, while the second sentence was generated by a statistical parser.

For the analyses in Figure 4.4, the LA metric would extract the paths listed in Table 4.4 for each terminal node in the trees. POS tags are not represented in the paths. Paths encode phrase boundaries, represented by square brackets. The following rules determine the insertion of a phrase boundary:

1. A left phrase boundary is inserted in the path of terminal node N immediately before the highest non-terminal symbol for which N is the leftmost child.

2. A right phrase boundary is inserted in the path of terminal node N immediately after the highest non-terminal symbol for which N is the rightmost child.

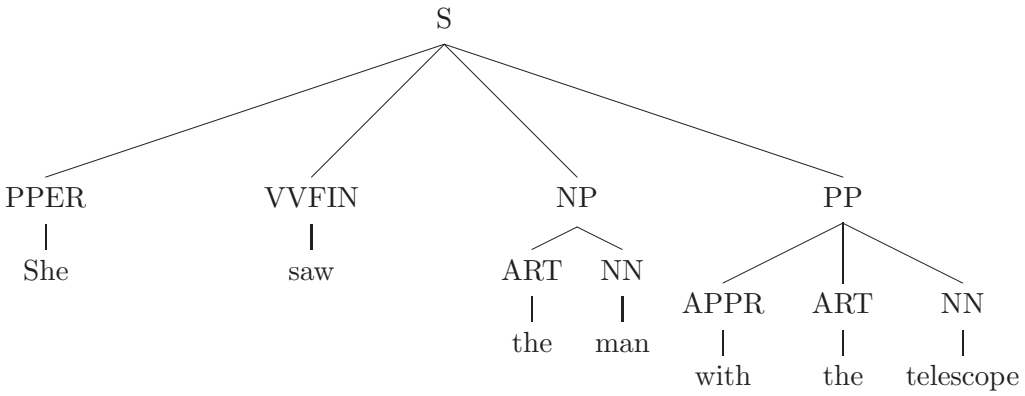
	TiGer	**TüBa-D/Z**	# errors
PP attachment I	97.5	99.2	593
PP attachment II	98.0	98.3	240
Label I	70.6	88.3	2851
Label II	92.5	97.0	725
Label III	95.9	98.4	399
SPAN I	99.4	99.8	57
SPAN II	97.9	99.1	208
total weighted ave.	81.6	92.6	5073

Table 4.3: F-score for PARSEVAL results for controlled error insertion in the original treebank trees

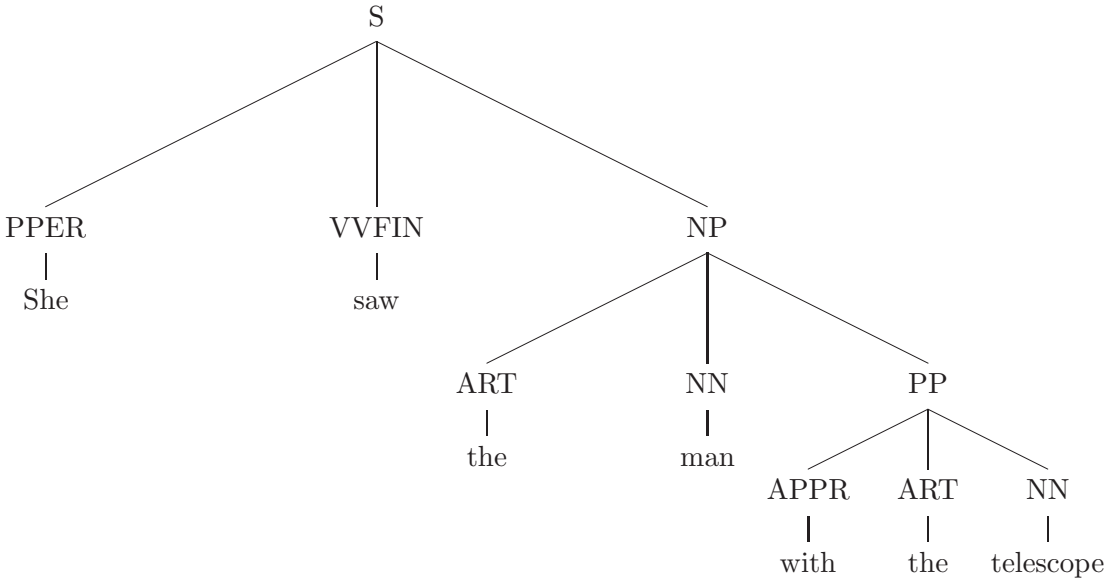
		gold paths		**parser output paths**
1.000	She	[S	:	[S
1.000	saw	S	:	S
1.000	the	[NP S	:	[NP S
0.800	man	NP] S	:	NP S
0.857	with	[PP S	:	[PP NP S
0.800	the	NP S	:	PP NP S
0.857	telescope	PP] S	:	PP NP] S
0.902	*average score*			

Table 4.4: LA paths and scores for example sentence in TiGer encoding

For the terminal node *She* the path consists of an opening bracket, according to the first rule, followed by the label S, and this is the same for gold tree and parser output. For the terminal node *saw* there is no non-terminal node for which *saw* is either the left-most or the right-most child node, so no phrase boundary is inserted. Therefore the path for *saw* consists of the label S only. The terminal *the* is the left-most child of the NP, so an opening bracket is inserted in the path right before the NP, which results in the path [NP S for both the gold tree and the parser output tree. For the first three terminal nodes the parser output paths are the same as the paths extracted from the gold trees and so they



(a) PP verb attachment



(b) PP noun attachment

Figure 4.4: Example sentences for PP attachment

receive a Levenshtein edit distance score of 1.0. The PP attachment ambiguity results in different paths for the remaining terminals. Again the score for each terminal is computed with the help of the Levenshtein edit distance, but with slight modifications. The Levenshtein edit distance assesses the similarity of two strings (s_1, s_2) by calculating the cost of converting s_1 into s_2. The cost for

each insertion, deletion or replacement required in the conversion process is 1. Therefore the basic function for computing the similarity of a gold path g and a parser output path p is described in (4.1).

$$1 - \frac{Lv(g,p)}{length(g) + length(p)} \tag{4.1}$$

However, the LA metric does a little bit more than that: the cost for each insertion or deletion is set to 1, but in order to distinguish between linguistically more or less severe errors the cost of replacing a node label in the path by another label is determined depending on the particular label. The cost of replacing two unrelated labels is set to 2, while replacing two labels closely related to each other incurs a cost of 0.5 only. Two labels are considered to be related if they start with the same character. As a result the LA metric gives worse results for a parse tree where an NM node (numerical node) has been falsely annotated as a PP than for a tree where the same node has been assigned an NP label.

In order to make use of this linguistically motivated feature, I transformed every PN node (proper name) in the TiGer treebank into the label NPN and every EN-ADD node (proper name) in TüBa-D/Z into NEN-ADD. I also converted all R-SIMPX nodes (relative clause) in TüBa-D/Z into the label SIMPX-R (in TiGer relative clauses are marked by the grammatical function label RC, so no conversion is needed). As a result the LA metric considers NP nodes and proper name nodes as well as simplex clauses and relative clauses as related and therefore punishes these errors less severely.

4.3.5 Comparing LA and PARSEVAL

Table 4.5 shows the results for the leaf-ancestor evaluation metric for the error insertion test sets (Section 4.3.2). The LA results for the two 1000 sentences test sets are much closer to each other than the corresponding PARSEVAL scores (92.2 vs. 95.5 as against 81.6 vs. 92.6). In fact, under the LA evaluation, only the label errors, due to the large numbers, show a significant difference between the two treebank annotation schemes.

	TiGer	**TüBa-D/Z**	# errors
PP attachment I	99.3	99.5	593
PP attachment II	99.3	99.0	240
Label I	87.8	92.3	2851
Label II	94.5	99.4	725
Label III	99.8	99.9	399
SPAN I	99.9	99.9	57
SPAN II	99.7	99.8	208
total weighted avg.	92.2	95.5	5073

Table 4.5: LA results for error insertion in the original treebank trees

To understand the difference between the two evaluation metrics, consider again the example sentences in Figure 4.4. PARSEVAL counts matching brackets in the gold tree and in the parser output. For the two sentences annotated according to the TiGer treebank encoding scheme, we obtain the following result:

(S She saw (NP the man) (PP with the telescope)) TiGer gold tree
(S She saw (NP the man (PP with the telescope))) Parser output
2 out of 3 brackets correct → **66.7%** *labelled F-score*

Now let us take the same sentences and annotate them according to the TüBa-D/Z encoding scheme. This time the result is different:

(S (VF (NP She)) (LK (VP saw)) (MF (NP the man**)** (PP with (NP the telescope)))
(S (VF (NP She)) (LK (VP saw)) (MF (NP the man (PP with (NP the telescope))**)**))
7 out of 8 brackets correct → **87.5%** *labelled F-score*

EVALB measures parser quality by counting matching brackets in the gold tree and the parser output. For the more hierarchical annotation scheme of the TüBa-D/Z, where the more deeply nested annotation results in a higher number of brackets for each tree, the effect of one mismatching bracket is substantially less severe than for TiGer. This shows that the PARSEVAL metric is biased towards annotation schemes with a high ratio of nonterminal vs. terminal nodes.

In contrast to this, the LA metric is less sensitive to the ratio of non-terminal vs. terminal nodes in the tree. Table 4.6 shows LA results for the same sentence in TüBa-D/Z encoding. While for PARSEVAL we observe a difference in scores between the two annotation schemes of more than 20%, LA results for the TüBa-D/Z-encoded sentence are only around 3% better than for TiGer. Table 4.6 shows that the same three terminals are affected by the error as for TiGer (Table 4.4), but due to the more hierarchical annotation and the extra layer of topological fields the paths in the TüBa-D/Z annotation scheme are longer than in TiGer. Therefore, the edit cost for inserting or deleting one symbol in the path, which is computed relative to path length, is lower for the TüBa-D/Z trees. This shows that the LA metric is also biased towards the TüBa-D/Z, but not to the same extent as the PARSEVAL metric.

		gold path		**parser output**
1.000	She	NP VF] [S	:	NP VF] [S
1.000	saw	VP [LK] S	:	VP [LK] S
1.000	the	NP [MF S	:	NP [MF S
0.857	man	NP] MF S	:	NP MF S
0.889	with	[PP MF S	:	NP MF S
0.909	the	[NP PP MF S	:	[NP PP NP MF S
0.909	telescope	NP PP MF S]	:	NP PP NP MF S]
0.938	*average score* for TüBa-D/Z			

Table 4.6: LA paths and scores for example sentence in TüBa-D/Z encoding

Experiment I showed that both PARSEVAL and (less so) the LA metric do favour treebank annotation schemes with a higher ratio of non-terminal versus terminal nodes in the tree, and thus do not provide a valid measure for cross-treebank evaluation. This means that the claim that German is not harder to parse than English (Kübler et al., 2006; Maier, 2006), which is based on a cross-treebank evaluation with PARSEVAL, does not hold.

4.4 Experiment II

Kübler (2005) and Maier (2006) assess the impact of the different treebank annotation schemes on PCFG parsing by conducting a number of modifications converting the TüBa-D/Z into a format more similar to the NEGRA (and hence the TiGer) treebank, essentially by flattening TüBa-D/Z trees. After each modification they extract a PCFG from the modified treebank and measure the effect of the changes on parsing results. They show that with each modification transforming the TüBa-D/Z into a more NEGRA-like format the parsing results also become more similar to the results of training on the NEGRA treebank, i.e. the results deteriorate. The authors take this as evidence that the TüBa-D/Z is more adequate for PCFG parsing. This assumption is based on the belief that PARSEVAL results fully reflect parse quality across treebanks and under different annotation schemes. This is not always true, as shown in the comparison between PARSEVAL and LA scores in Experiment I (Section 4.3.5).

In the second experiment I crucially change the order of events in the Kübler (2005), Kübler et al. (2006) and Maier (2006) conversion experiments: I first extract an unlexicalised PCFG from each of the original treebanks. I then transform the output of the parser trained on the TüBa-D/Z into a format more similar to the TiGer treebank. In contrast to Kübler (2005), Kübler et al. (2006) and Maier (2006), who converted the treebank *before extracting the grammars* in order to measure the impact of single features like topological fields or unary nodes on PCFG parsing, I convert the trees in the parser *output* of a parser trained on the *original* unconverted treebank resources. This allows me to preserve the basic syntactic structure and also the errors present in the output trees resulting from a potential bias in the original treebank training resources. The expectation is that the results for the original parser output evaluated against the unmodified gold trees should not be crucially different from the results for the modified parser output evaluated against the modified gold trees. If this is not the case, then the outcome is further evidence that different encodings react differently to what are the same parsing errors and again we cannot conclude that German is not harder to parse than English.

4.4.1 Experimental Setup

For Experiment II I trained BitPar (Schmid, 2004), a statistical parser for highly ambiguous PCFG grammars, on the two treebanks. The TüBa-D/Z training data consists of the 26125 treebank trees not included in the TüBa-D/Z test set. Because of the different size of the two treebanks I randomly selected 26125 sentences from the TiGer treebank (excluding the sentences in the TiGer test set).

Before extracting the grammars I resolved the crossing branches in the TiGer treebank by attaching the non-head child nodes higher up in the tree, following Kübler et al. (2006). As a side-effect this leads to the creation of some unary nodes in the TiGer trees. I also inserted a virtual root node in the TiGer and TüBa-D/Z data sets and removed all functional labels from the trees. After this preprocessing step I extracted an unlexicalised PCFG from each of the training sets. The TiGer grammar has a total of 24504 rule types, while the grammar extracted from the TüBa-D/Z treebank consists of 5672 rules only. I parsed the TiGer and TüBa-D/Z test set with the extracted grammars, using raw text for parser input. Then I automatically converted the TüBa-D/Z-trained parser output to a TiGer-like format and compared the evaluation results for the unmodified parser output trees against the original gold trees with the results for the converted parser output against the converted gold trees.

4.4.2 Converting the TüBa-D/Z Trees to TiGer-Style Trees

The automatic conversion of the TüBa-D/Z-style trees includes the removal of topological fields and unary nodes as well as the deletion of NPs inside of PPs, because the NP child nodes are directly attached to the PP in the TiGer annotation scheme. As a last step in the conversion process I adapted the TüBa-D/Z node labels to the TiGer categories.

4.4.3 The Conversion Process: A Worked Example

I demonstrate the conversion process using an example sentence from the TüBa-D/Z test set (TüBa-ORIG) ((8) and Figure 4.5). Topological fields, here VF

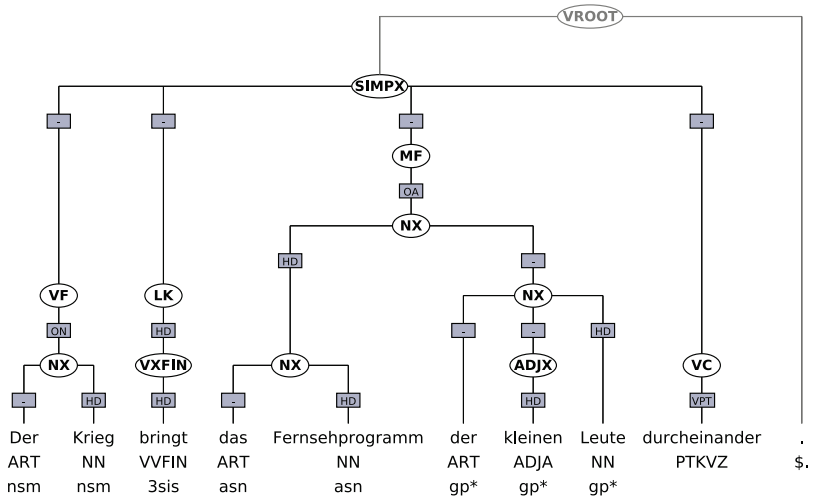


Figure 4.5: Original TüBa-D/Z-style gold tree

(initial field), MF (middle field), LK (left sentence bracket) and VC (verb complex), as well as unary nodes are removed. The category labels have been changed to TiGer-style annotation. The converted tree (TüBa-ORIG-CONV) is given in Figure 4.6.

(8) Der Krieg bringt das Fernsehprogramm der kleinen Leute
The war messes the TV program (of) the little people
durcheinander.
about.
"War messes about the TV program of ordinary people."

Figure 4.7 shows the unmodified parser output from the TüBa-D/Z-trained parser (TüBa-PARSE) for the same string. The parser incorrectly attached the two NPs directly to the middle field, while in the gold tree (Figure 4.5) both NPs are attached to an NP which is a child node of the middle field. The TiGer-style modified parser output (TüBa-PARSE-CONV) is shown in Figure 4.8.

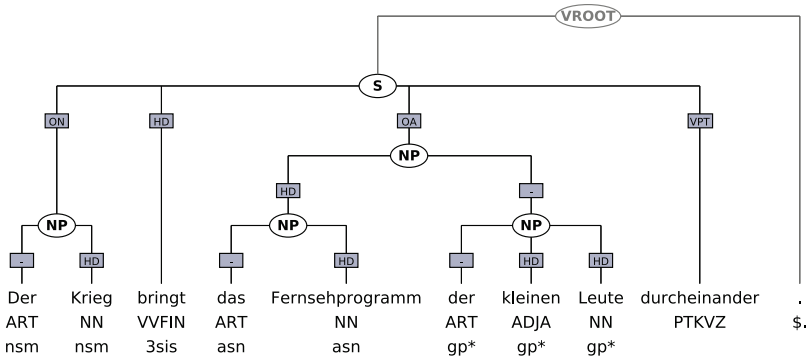


Figure 4.6: Converted TüBa-D/Z to TiGer-style gold tree

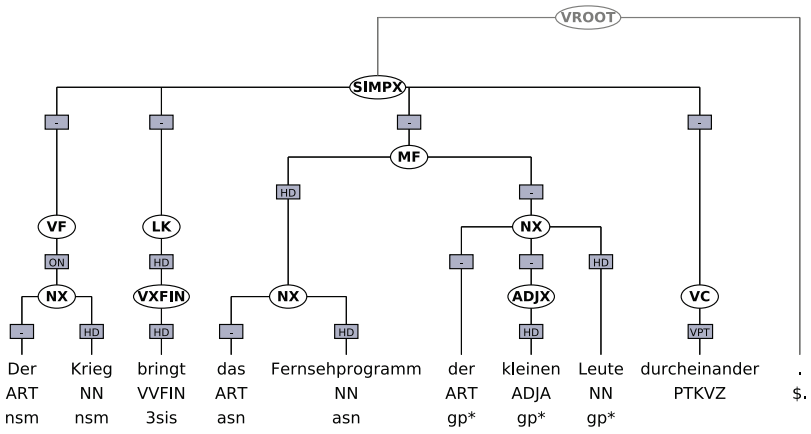


Figure 4.7: Parser output (trained on TüBa-D/Z)

4.4.4 Results for Converted Parser Output

I applied the conversion method described above to the original TüBa-D/Z trees and the TüBa-D/Z-trained parser output for the sentences in the TüBa-D/Z test set. Table 4.7 shows PARSEVAL and LA results for the modified trees, evaluating the (converted) parser output for each treebank against the (converted) gold trees of the same treebank, using gold POS tags as parser input (results for raw text are given in Table 4.8). Due to the resolved crossing branches in the

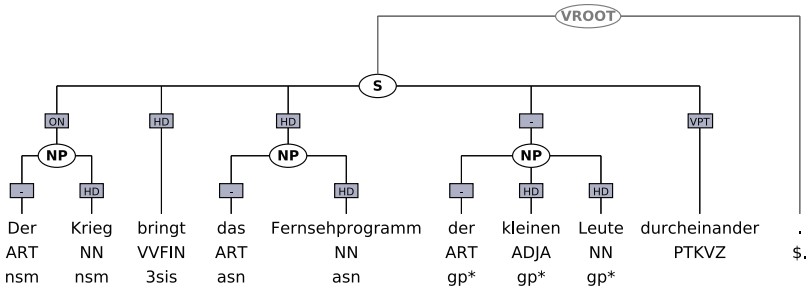


Figure 4.8: TüBa-D/Z to TiGer-style converted parser output

TiGer treebank we also have some unary nodes in the TiGer test set. Their removal surprisingly improves both PARSEVAL and LA results.[7]

Table 4.7 shows that for the TüBa-D/Z, all conversions lead to a decrease in F-score for the PARSEVAL metric. However, precision improves slightly when removing topological fields from the TüBa-D/Z trees. For the LA metric the flattening of PPs improves the average score.

After applying all conversion steps to the data and thereby effectively converting the trees parsed by the TüBa-D/Z grammar to a TiGer-like format, we observe a PARSEVAL F-score for the TüBa-D/Z test set which is lower than that for the TiGer trees. The LA metric gives better results for the original TiGer trees compared to the result for the unmodified TüBa-D/Z trees. Here the treebank modification has no strong effect on parsing results.

Table 4.8 shows results for the same experimental setting, this time using raw

[7]This is caused by the fact that both measures compute scores relative to the overall number of brackets in the tree and path length, respectively. Example 9 illustrates this. The example shows a sentence from the gold standard (9), including a unary VP node *(VP (VVPP geleugnet))*. The parser output tree for this sentence is exactly the same for both settings, with and without unary nodes. First we evaluate the parser output sentence against the gold standard sentence with the unary node and get an `evalb` score of 66.67 for both, precision and recall (see table below).

(9) (S (PP (APPR In) (ART dem) (NN Pamphlet)) (VAFIN wird) (NP(ART die)
In the pamphlet becomes the
(NN Judenvernichtung) (PP (APPR in) (NE Auschwitz))) (VP (VVPP geleugnet))))
holocaust in Auschwitz denied

"The pamphlet denies the holocaust in Auschwitz"

Gold POS tags as parser input					
	prec.	**recall**	**F-sco.**	LA	
TiGer	78.4	77.2	77.8	93.6	*TiGer-PARSED*
no Unary	78.5	77.8	78.2	93.6	*against*
					TiGer-ORIG
TüBa-D/Z	89.3	83.9	86.5	92.0	*TüBa-PARSED*
					against
					TüBa-ORIG
TüBa-D/Z → TiGer					
no Topological	89.3	82.3	85.7	91.5	*TüBa-PARSED-CONV*
no Unary	83.7	76.4	79.9	91.3	*against*
no Top + no Unary	83.4	74.0	78.4	90.6	*TüBa-ORIG-CONV*
no Top + no Unary	80.1	71.8	75.7	91.2	
+ flatten PPs					

Table 4.7: The impact of the conversion process on PARSEVAL and LA (gold POS)

text as parser input. For TiGer, results for perfect tags (77.8% F-score) and for raw text (76.7% F-score) are quite close, while for TüBa-D/Z the use of gold POS tags has a more profound effect and leads to an increase in F-score of around 3%.

		Sent.			Matched	Bracket	
	ID	Length	Recal	Prec.	Bracket	gold	test
unary	1	10	66.67	66.67	4	6	6
no unary	1	10	80.00	66.67	4	5	6

For the same parser output tree evaluated against the gold standard tree without the unary node, we obtain a precision of 66.67 and a recall of 80.00 (see Table above, *no unary*). This is due to the fact that the gold tree without unary nodes has one pair of brackets less than the one with the unary node. As a result the number of matching brackets in the parser output tree and gold standard is divided by 5, not by 6, as was the case for the gold tree including the unary node. Unary nodes mostly occur in the gold standard, but not so much in the parser output. Thus results for parser output trees improve when removing unary nodes from the gold standard.

Raw text as parser input					
	prec.	**recall**	**F-sco.**	LA	
TiGer	77.3	76.1	76.7	93.2	*TiGer-PARSED*
no Unary	77.4	76.8	77.1	93.3	*against*
					TiGer-ORIG
TüBa-D/Z	86.4	81.0	83.6	91.1	*TüBa-PARSED*
					against
					TüBa-ORIG
TüBa-D/Z → TiGer					
no Topological	86.6	79.5	82.9	90.8	*TüBa-PARSED-CONV*
no Unary	81.5	74.4	77.8	90.5	*against*
no Top + no Unary	81.9	72.3	76.8	90.0	*TüBa-ORIG-CONV*
no Top + no Unary	78.6	70.0	74.0	90.6	
+ flatten PPs					

Table 4.8: The impact of the conversion process on PARSEVAL and LA (raw text)

When parsing raw text we observe the same trend in the results for the conversion process as we did when using gold POS tags.

The constant decrease in PARSEVAL results for the modified trees is consistent with the results in Kübler et al. (2006) and Maier (2006), but my conclusions are crucially different. Experiment II shows that the decrease in parsing results reported in Kübler et al. (2006) and Maier (2006) does not reflect a decrease in parser output quality, as in my experiment the original parser output and the converted parser output trees contain the same basic structure and, crucially, the same parsing errors. The lower results for the converted parser output are due to the sensitivity of the PARSEVAL metric to the TiGer/TüBa-D/Z data structures, in particular the ratio of non-terminal vs. terminal nodes in the trees.

4.5 Experiment III

Experiments I and II show that the tree-based PARSEVAL metric does not provide a reliable measure for comparing the impact of different treebank annotation schemes on the quality of parser output and so the question whether German is harder to parse than English is still undecided. In Experiment III I present a dependency-based evaluation and compare the results to the results of the two constituency-based evaluation metrics, PARSEVAL and LA.

4.5.1 Dependency-Based (DB) Evaluation

The dependency-based evaluation used in the experiments follows the method of Lin (1998) and Kübler and Telljohann (2002), converting the original treebank trees and the parser output into bilexical POS-labelled dependency relations of the form WORD POS HEAD. Functional labels have been omitted for parsing, so the dependencies do not comprise functional information.[8]

Figure 4.9 shows the CFG representation in the TiGer treebank style for the gold tree in Figure 4.4 (a). Square boxes denote grammatical functions. Figure 4.10 shows the dependency relations for the same tree, indicated by labelled arrows. Converted into a WORD POS HEAD triple format the dependency tree looks as in Table 4.9.

I assessed the quality of the automatic dependency conversion methodology by converting the 1000 original trees from each of the test sets into bilexical, POS-labelled dependency relations. In TiGer, verbal heads are annotated with the label HD, so for the personal pronoun *She* in Figure 4.9 the head is the sister node with label HD, *saw*, which results in the dependency relation `She` PPER `saw`. Unfortunately TiGer does not annotate the lexical heads of PPs and NPs, which makes it necessary to use heuristic head-finding rules for the dependency conversion.

[8] Note that the bilexical POS-labelled dependency relations are different from labelled dependency triples using grammatical functions, as POS labels do not specify grammatical relations between a head and its dependent.

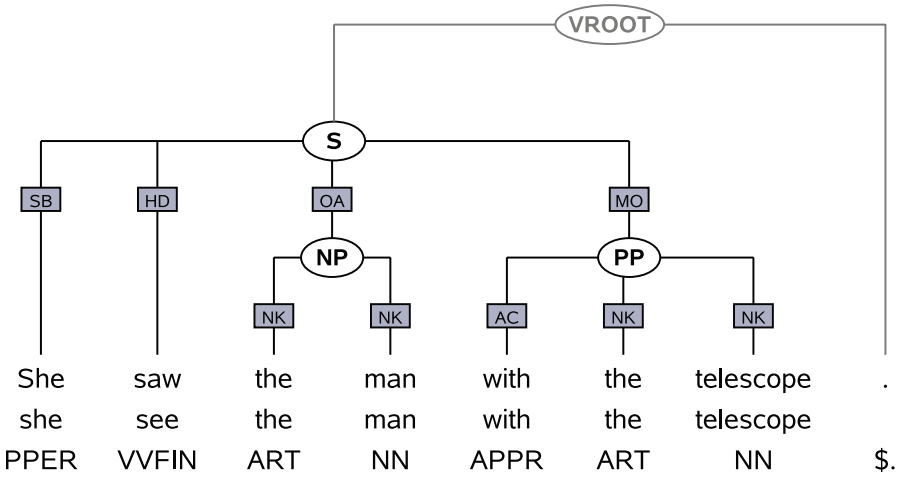


Figure 4.9: TiGer treebank representation for Figure 4.4 (a) (page 45)

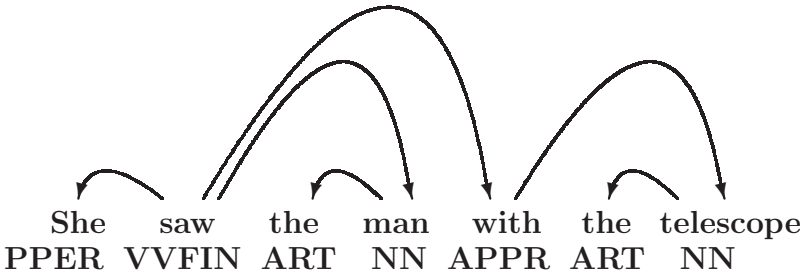


Figure 4.10: Dependency tree for Figure 4.9

After converting the original trees into dependencies, using the grammatical function labels to support the evaluation, I then removed all functional information from the original treebank trees and converted the stripped trees into dependencies, using heuristics to find the head of each node. I evaluated the dependencies for the stripped gold trees against the dependencies for the original gold trees including functional labels and obtained an F-score of 99.65% for TiGer and 99.13% for the TüBa-D/Z dependencies. This shows that the conversion is reliable and not unduly biased to either the TiGer or TüBa-D/Z annotation

WORD	POS	HEAD
She	PPER	saw
saw	VVFIN	-
the	ART	man
man	NN	saw
with	APPR	saw
the	ART	telescope
telescope	NN	with

Table 4.9: Dependency triples for Figure 4.9

schemes.

4.5.2 Experimental Setup

For Experiment III I used the same PCFG grammars and test sets as in Experiment II. I used both raw text and gold POS tags as parser input.

4.5.3 Results

Table 4.10 shows the evaluation results for the three evaluation metrics using gold POS tags (we repeat results for the constituency-based evaluation from Section 4.4.4). For the dependency-based evaluation the parser trained on the TiGer training set achieves significantly higher results for precision and recall than the parser trained on the TüBa-D/Z. This is clearly in contrast to the PARSEVAL scores, which show higher precision and recall for the TüBa-D/Z. Table 4.11 shows the same trends for parsing raw text. In contrast to the PARSEVAL results on gold POS tags (Table 4.10), the gap between the PARSEVAL results for TiGer and TüBa-D/Z parsing raw text (Table 4.11) is not as wide as before.

The considerable difference between the results for the different evaluation methods raises the question as to which of the metrics is the most adequate for judging parser output quality. In Chapter 5 I will return to this question by comparing automatic evaluation results with human judgements.

Gold POS tags as parser input						
	Dependencies		PARSEVAL			LA
	Prec	Rec	Prec	Rec	F-sco.	avg.
TiGer	88.2	88.3	78.4	77.2	77.8	93.6
TüBa-D/Z	76.6	76.6	89.3	83.9	86.5	92.0

Table 4.10: Parsing results for three evaluation metrics (gold POS)

Raw text as parser input						
	Dependencies		PARSEVAL			LA
	Prec	Rec	Prec	Rec	F-sco.	avg.
TiGer	83.1	83.1	77.3	76.1	76.7	93.2
TüBa-D/Z	76.6	76.6	86.4	81.0	83.6	91.1

Table 4.11: Parsing results for three evaluation metrics (raw text)

4.5.4 Related Work

Boyd and Meurers (2008) present a labelled dependency evaluation based on PCFG parser output of the LoPar parser (Schmid, 2000) trained on the NEGRA and TüBa-D/Z treebanks. They point out that the evaluation of Kübler et al. (2006) did not consider grammatical function labels attached to terminal nodes, which means that a substantial part of the GF labels in the NEGRA treebank were not included in the evaluation. Boyd and Meurers provide an evaluation for the main grammatical functions and give results for all subjects, accusative objects and dative objects, regardless of whether the underlying label was attached to a terminal or non-terminal argument. They report better labelled dependency F-scores for all three grammatical functions for the parser trained on the NEGRA treebank compared to the parser trained on TüBa-D/Z (Table 4.12). This result is in contrast to the results of Kübler et al. (2006), and provides further evidence for my claim that PARSEVAL is not a meaningful measure for parser evaluation across treebanks.

	NEGRA			TüBa-D/Z		
	Prec	Rec	F-sco.	Prec	Rec	F-sco.
Subj	69.7	69.1	69.4	65.7	72.2	69.0
Acc	48.2	51.0	49.6	41.4	46.8	44.1
Dat	20.9	15.2	18.1	21.4	11.5	16.5

Table 4.12: Labelled dependency F-scores (Boyd and Meurers, 2008) for main GFs in NEGRA and TüBa-D/Z

4.6 Conclusions

In this chapter I presented experiments assessing the validity of parsing results measured along different dimensions: the tree-based PARSEVAL metric, the string-based Leaf-Ancestor metric and a dependency-based evaluation. By inserting controlled errors into gold treebank trees and measuring the effects on evaluation results, I gave new evidence for the problems of using PARSEVAL which, despite severe criticism, is still the standard measure for PCFG parser evaluation. I showed that PARSEVAL cannot be used to compare the output of PCFG parsers trained on different treebank annotation schemes, because PARSEVAL results correlate with the ratio of non-terminal/terminal nodes in the trees. Comparing two different annotation schemes, PARSEVAL consistently favours the one with the higher node ratio.

I examined the influence of treebank annotation schemes on unlexicalised PCFG parsing, and rejected the claim that the German TüBa-D/Z treebank is more appropriate for PCFG parsing than the German TiGer treebank. I showed that converting the TüBa-D/Z parser output to a TiGer-like format leads to PARSEVAL results which are slightly worse than the ones for the TiGer treebank. Additional evidence comes from a dependency-based evaluation, showing that, for the output of the parser trained on the TiGer treebank, the mapping from the CFG trees to dependency relations yields better results than for the grammar trained on the TüBa-D/Z annotation scheme, even though PARSEVAL scores suggest that the TiGer-based parser output trees are substantially worse than TüBa-D/Z trees. This means that contrary to Kübler et al. (2006), the

question whether German is harder to parse than English or not is still undecided. Future work might explore the impact of automatic controlled error insertion and cross-treebank conversion on results of the dependency-based evaluation.

The experiments presented in this chapter showed that the PARSEVAL metric does not support a meaningful cross-treebank comparison. In the next chapter I discuss other pitfalls for cross-treebank evaluation, such as out-of-domain problems or differences in linguistic analysis between different treebanks.

Chapter 5

TiGer and TüBa-D/Z: Apples and Oranges

5.1 Introduction

In the last chapter I showed that neither PARSEVAL nor the Leaf-Ancestor metric are valid measures for cross-treebank comparisons, which raises the question how to perform a fair and unbiased comparison of treebanks (and resources derived from these treebanks) with different encoding schemes and, at the same time, avoid comparing apples with oranges.

There are a number of attempts, based on statistical measures, to compare syntactic structure in different corpora: Nerbonne and Wiersma (2006) present an aggregate measure of syntactic distance based on POS trigrams. Sanders (2007) uses Leaf-Ancestor path-based permutation tests to measure differences between dialectal variations of British English. (Corazza et al., 2008) describe a measure based on conditional cross-entropy to predict parser performance for a parser trained on different treebanks. Out of the studies mentioned above the last one is the closest to our interests. However, in contrast to Corazza et al., who aim at developing a measure to assess the parseability of different corpora, we aim at obtaining detailed knowledge about the pros and cons of specific treebank design decisions and their impact on parser performance.

The next sections provide a thorough comparison of two German treebanks, the TiGer treebank and the TüBa-D/Z. I use simple statistics on sentence length

and vocabulary size, and more refined methods such as perplexity and its correlation with PCFG parsing results, as well as a Principal Component Analysis. I also investigate the impact of sampling methods on comparisons. After discussing the differences between the two corpora I present a qualitative evaluation of a set of 100 sentences from the TüBa-D/Z, manually annotated in the TiGer as well as in the TüBa-D/Z annotation scheme, and show that even the existence of a parallel subcorpus does not support a straightforward and easy comparison of both annotation schemes.

Part of the research presented in this chapter has been published in Rehbein and van Genabith (2007b).

5.2 Comparing the Treebanks

For the experiments I divided both treebanks into sets of samples without replacement with 500 sentences each, randomly selected from the two treebanks, which resulted in 100 samples for the TiGer treebank and 44 samples for the TüBa-D/Z. In order to account for the different size of the treebanks I used samples 1-44 from the TüBa-D/Z treebank as well as samples 1-44 (TiGer1) and 45-88 (TiGer2) from the TiGer treebank.

As I am interested in the influence of sampling techniques on parsing results I also generated a second set of samples with 500 trees each, which were taken in sequential order from the treebanks (rather than randomly as in the first set described above). This means that, in contrast to the random samples, the content in each sample is "semantically" related, which most obviously must have a crucial impact on vocabulary size and homogeneity of the samples.

5.2.1 Sentence Length / Word Length / Vocabulary Size

The average sentence length in TiGer is comparable to the one in TüBa-D/Z (Table 5.1), but the average word length in TüBa-D/Z is shorter than in TiGer. TüBa-D/Z also uses a smaller vocabulary than the TiGer treebank, which is most probably due to the shorter period of time covered by the articles in the

corpus.[9] (Stylistic differences between the two newspapers may also have an impact on vocabulary size, see Section 5.2.2). As noted previously, due to the flat annotation in TiGer the ratio of non-terminal vs. terminal nodes is much smaller than in TüBa-D/Z. While the treebanks are comparable with regard to text domain and sentence length, there are considerable differences concerning word length and vocabulary size between the two corpora. In the next section I investigate the distribution of POS tags in TiGer and TüBa-D/Z, using Principal Component Analysis.

	avg. sent. length (rand)	**avg. word length (rand)**	**avg. vocab size (rand)**	**avg. vocab size (seq)**	**non-term. /terminal**
TiGer1	17.86	6.27	2992	2638	0.47
TiGer2	17.03	6.27	2989	2662	0.47
TüBa-D/Z	17.25	5.70	2906	2585	1.20

Table 5.1: Some properties of the TiGer and TüBa-D/Z treebank

5.2.2 Principal Component Analysis (PCA) of POS Tags

PCA is a way of reducing complex, high-dimensional data and detecting underlying patterns by transforming a high number of (possibly) correlated variables in a multivariate data set into a smaller number of uncorrelated variables whilst retaining as much as possible of the variation present in the data. The uncorrelated new variables are called principal components or *eigenvectors*. They are chosen in such a way that high correlating variables are combined into a new variable which describes the largest part of the variance in the data. The new variable constitutes the first principal component. Next the second component is chosen so that it describes the largest part of the remaining variance, and so on. PCA has been successfully applied to a number of tasks such as the analysis of register variation (Biber, 1998) or authorship detection (Juola & Baayen, 1998).

Figure 5.1 shows the 1st and 2nd components of a PCA based on the frequency counts of POS tags in the randomised samples, which together capture around

[9]The TiGer treebank (Release 2) contains newspaper articles from 1992/1994, while the TüBa-D/Z (Release 2) covers a period of one month only (May 1999).

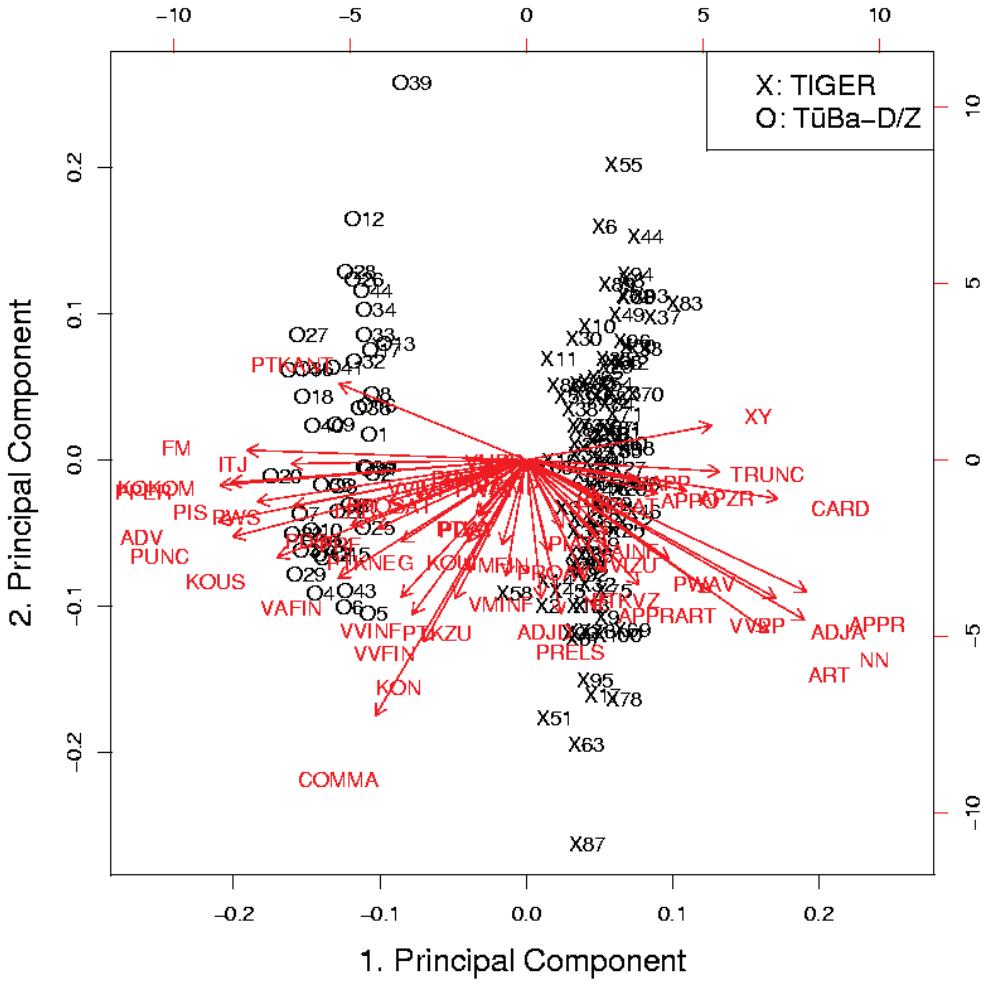


Figure 5.1: PCA for TiGer/TüBa-D/Z POS tags

33% of the variance in the data. The first component clearly separates TiGer from TüBa-D/Z samples. TüBa-D/Z is characterised by a high number of informal elements such as interjections, foreign language material (mostly Anglicisms), indefinite and interrogative pronouns and indicators of a personal style such as personal pronouns. TiGer samples show a high number of nouns, determiners, attributive adjectives, prepositions and also circumpositions, past participles and first elements of compounds. A high number of nominal elements (nouns, compounds, nominalised adjectives) is typical for a nominative style (Ziegler et al., 2002), which is often interpreted as being more objective and informative than a

verbal style. I tend to interpret the first component as a dimension of informality, where formal texts with a high degree of information content are positioned at one end and informal texts written in a more personal and subjective style at the other end.

5.2.3 Perplexity

Kilgariff (2001) describes how the information-theoretic measure of *cross-entropy* can be used to assess the *homogeneity* of a text corpus. Perplexity is the log of the cross-entropy of a corpus with itself and can be interpreted as a measure of *self-similarity* of a corpus: the higher the perplexity, the less homogeneous the corpus. Perplexity can be unpacked as the inverse of the corpus probability, normalised by corpus size (5.1).

$$PP(W) = P(w_1 ... w_N)^{\frac{1}{N}} = \sqrt[N]{\Pi^N{}_{i=1} \frac{1}{P(w_i|w_{1...i-1})}} \tag{5.1}$$

I compute the perplexity for language models derived from each of the treebanks.[10] As I am mostly interested in parsing results it is questionable whether a simple word trigram model provides the information I am looking for. Hence I also computed perplexity for a POS trigram model and for a trigram model based on Leaf-Ancestor (LA) paths (Sampson & Babarczy, 2003). LA measures the similarity of the path of each terminal node in the parse tree to the root node. The path consists of the sequence of node labels between the terminal node and the root node, and the similarity of two paths is calculated by using the Levenshtein distance (Levenshtein, 1966). For a more detailed description see Chapter 4.3.4. I assume that POS trigrams and LA path representations are more adequate to approximate the syntactic structure of a sentence and to allow predictions about parsing results.[11]

[10] The language models were produced and calculated using the CMU/Cambridge toolkit (`http://mi.eng.cam.ac.uk/~prc14/toolkit.html`)

[11] Note that the LA-path-based representations used for generating the language models do not include grammatical functions.

I report experiments on both the randomised and sequential samples. For TüBa-D/Z we have a total of 44 samples with 500 trees each in a 44-cross-validation-style experiment. I compute the perplexity for each of the 44 samples by training a language model on the remaining 43 samples and testing the model on the held-out sample. For TiGer1 and TiGer2 I proceeded as described for TüBa-D/Z.

Table 5.1 shows that the "semantic relatedness" in the sequential samples has a crucial impact on the size of the vocabulary. I expect that this will lead to a higher predictability of the structure in the sequential samples compared to the randomised samples, which should result in a lower perplexity for the sequential samples. I also expect that, due to the smaller vocabulary in the TüBa-D/Z, perplexity for the TüBa-D/Z samples will be lower than for the TiGer samples. Table 5.2 shows results for all samples.

	sequential			*randomised*		
	word trigram	**POS trigram**	LA **path**	**word trigram**	**POS trigram**	LA **path**
TiGer1	599	8.8	6.0	681	8.9	6.1
TiGer2	643	8.8	5.9	684	8.9	6.0
TüBa-D/Z	665	9.4	4.3	651	9.4	4.3

Table 5.2: Perplexity (word/POS/LA-path-based trigram model) for TiGer and TüBa-D/Z

As expected, perplexity for the randomised TiGer samples is slightly higher than for the samples taken in sequential order from the corpus. For TüBa-D/Z, however, perplexity for the sequential word trigram model is higher than for the randomised samples. There is no such effect of "semantic relatedness" on syntactic homogeneity in the TüBa-D/Z. This again might be due to the fact that the TüBa-D/Z samples cover a smaller period in time and so the overall variance between the samples is lower than in TiGer. While this assumption is supported by the lower perplexity for the randomised word trigram model, it is all the more surprising that the perplexity for the TüBa-D/Z, computed for a POS trigram model, is so much higher than for the TiGer samples. This suggests that,

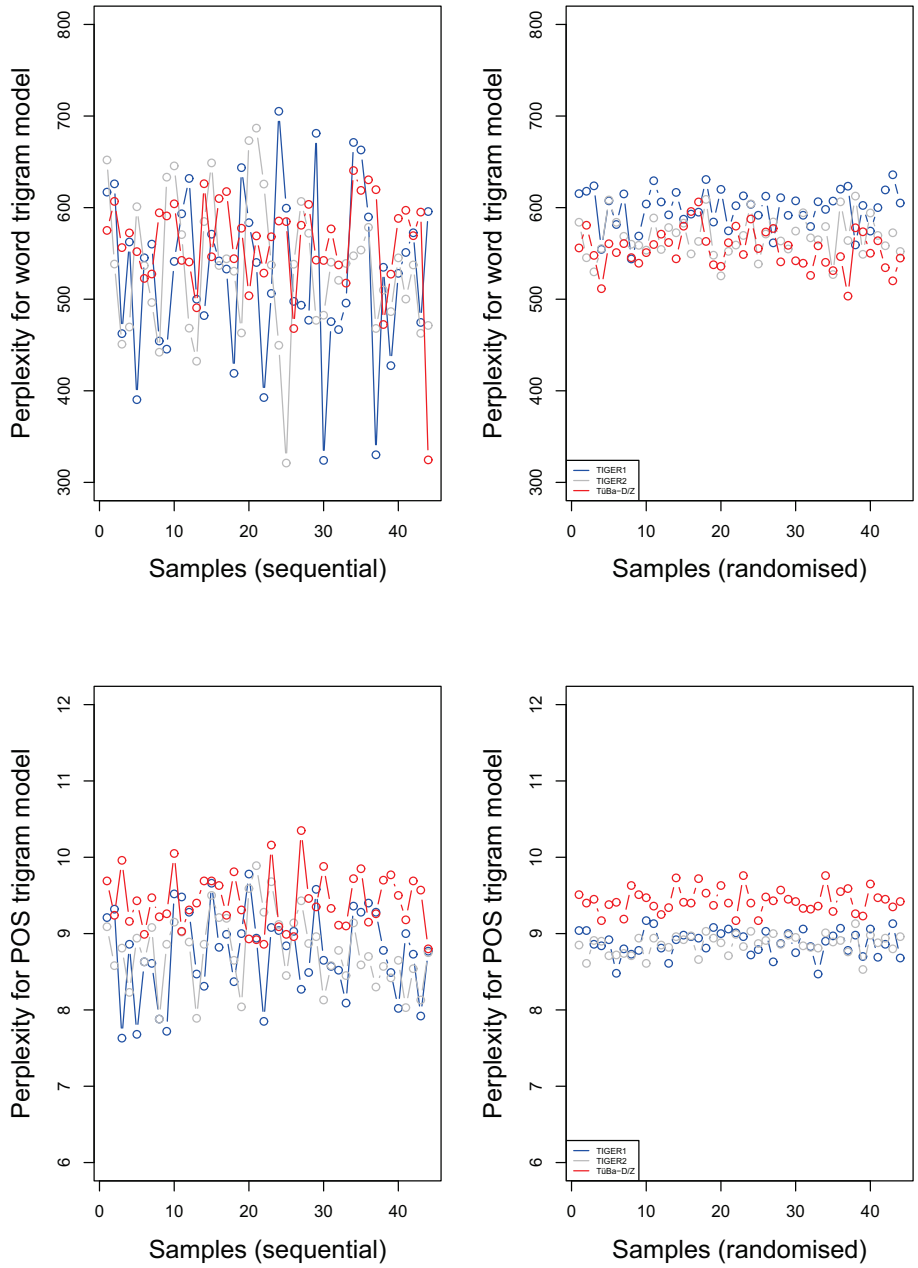


Figure 5.2: Perplexity for randomised and sequential samples (word/POS trigram model)

despite having text from the same domain (newspaper text), there are crucial differences between the structural properties of the texts in the two corpora.

Figure 5.2 shows the perplexity for the word and POS trigram models (sequential and randomised) for each sample in TiGer and TüBa-D/Z. It can be seen that, while the averaged results for the POS trigram models for the sequential and randomised samples are close or even identical, variation between results is much higher for the sequential samples. It can also be seen that for the sequential word trigram models, the variation between the TiGer samples is much higher than between the samples taken from the TüBa-D/Z, which again might be an effect of the larger period in time covered by the TiGer samples.

Results for the LA-path-based models diverge from the POS trigram model: despite its smaller vocabulary size, the POS-trigram perplexity indicates that the syntactic structure in the TüBa-D/Z is less homogeneous than in TiGer, and hence expected to be harder to parse. By contrast, the LA-path-based perplexity shows that TiGer (and crucially its annotation scheme as captured by the LA-path-based perplexity) is less homogeneous than TüBa-D/Z. In order to resolve this puzzle, in the next section I will investigate the correlation between (POS- and LA-path-based) perplexity and PCFG parsing results.

5.2.4 Parsing Experiments

For the parsing experiments I trained the PCFG parser BitPar (Schmid, 2004) on the data sets in 44-fold cross-validation-style experiments. For each sample, the training data consists of all remaining samples, so for the first TüBa-D/Z sample I trained the parser on samples 2-44, for sample 2 on samples 1 and 3-44 of the treebank, and so forth; and similarly for TiGer1 and TiGer2. In the experiments described below I used raw text as parser input.

Preprocessing

Before extracting the grammars, following Kübler (2005) I resolved the crossing branches in TiGer by attaching the non-head child nodes higher up in the tree and, where grammatical function labels such as *subject* or *accusative object* were directly attached to the terminal node, I inserted an additional unary node to

prevent the POS tagset for the TiGer grammar from being blown up artificially. The node insertion increases the ratio of non-terminal vs. terminal nodes in the TiGer treebank from 0.47 to 0.5 (compared to 1.2 in TüBa-D/Z). Figure 5.3 illustrates the insertion of preterminal nodes.

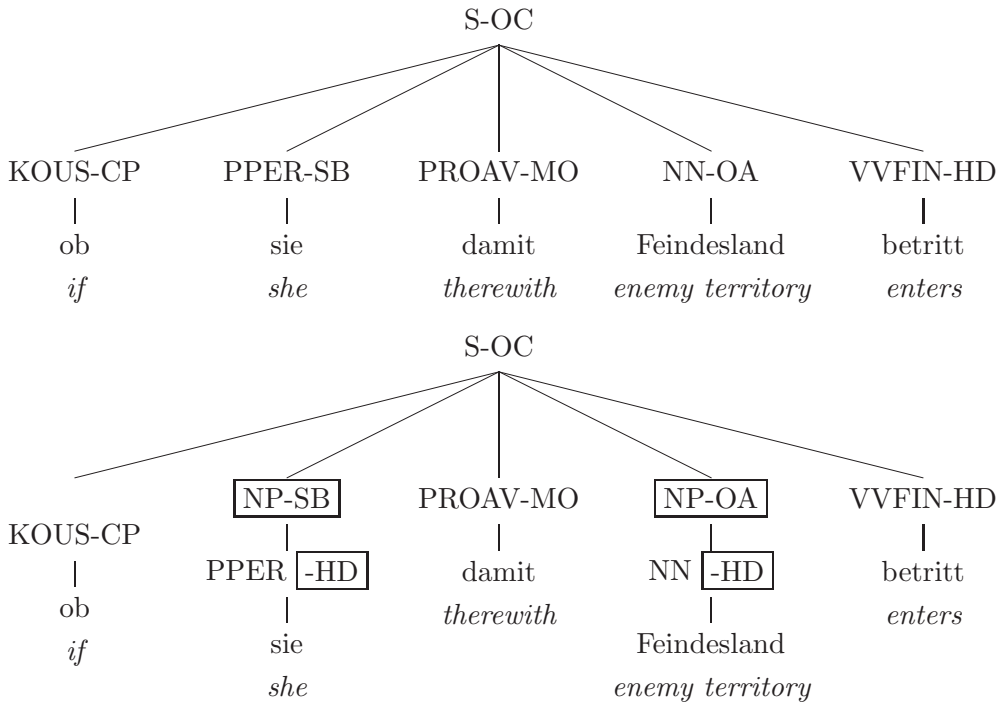


Figure 5.3: Preprocessing for TiGer: insertion of preterminal nodes

I then extract a PCFG from each of the training sets and parse the test sets. I evaluate parsing results using EVALB (results report labelled bracketing F-score), an implementation of the PARSEVAL metric, as well as the Leaf-Ancestor (LA) metric (Sampson and Babarczy, 2003).

Results

Table 5.3 shows averaged EVALB and Leaf-Ancestor (LA) results for the randomised and the sequential samples in the test sets. For all three data sets the EVALB results for the randomised samples show less variation (min. 71.5 and max. 76.5 for TiGer; min. 80.9 and max. 84.1 for TüBa-D/Z), while the results for the

sequential samples are distributed over a wider range from 70 to 79.2 for TiGer and 78 to 85.8 for TüBa-D/Z. EVALB gives around 10% better results for the parser trained and evaluated on the TüBa-D/Z, while the LA results are much closer across the treebanks within the 88-89% range. Table 5.3 also shows that the rankings given by EVALB and LA do not necessarily correlate: while for TiGer1 and TüBa-D/Z LA gives better results for the sequential samples, EVALB ranks the randomised samples as the ones with the higher quality in parser output.[12]

In Chapter 4 I showed that the remarkable difference in EVALB results for TiGer and TüBa-D/Z reflects the different data structures in the two treebanks and that EVALB cannot be used for cross-treebank comparisons. Therefore I now focus on the correlation between parser performance and perplexity for each parsing model (Table 5.4).

For the POS trigram model I compute a strong correlation between perplexity and LA as well as EVALB parsing results for sequential TiGer samples and a weak correlation for sequential TüBa-D/Z samples. By contrast, the LA-path-based trigram model shows a strong correlation for TiGer and TüBa-D/Z samples. For both models there is no correlation for randomised samples. This means that while for sequential samples a higher perplexity corresponds to lower EVALB and LA results, this observation does not hold for randomised samples. The same is true for sentence length: while there is a negative correlation between sentence length and parsing results for TiGer samples and, to a lesser extent, for TüBa-D/Z, for randomised samples there is a weak correlation of around -0.45 only. This shows that randomisation succeeded in creating representative samples, where the variation between training and test samples is not high enough

[12]Note that the differences between results are small and may not be statistically significant.

		TiGer1	**TiGer2**	**TüBa-D/Z**
LA (avg.)	*sequential*	88.36	88.45	89.14
	randomised	88.21	88.49	88.95
EVALB	*sequential*	74.00	73.45	82.80
($\leq$ 40)	*randomised*	74.33	74.00	83.64

Table 5.3: avg. LA and EVALB results for TiGer and TüBa-D/Z samples

	Perplexity/LA		Perplexity/EVALB		sent. length/	
	POS-n-gram	LA-path	POS-n-gram	LA-path	LA	EVALB
TiGer1	-0.89	-0.87	-0.76	-0.78	-0.80	-0.78
TiGer2	-0.81	-0.93	-0.81	-0.87	-0.89	-0.81
TüBa-D/Z	-0.47	-0.81	-0.49	-0.74	-0.73	-0.60

Table 5.4: Pearson's product-moment correlation (sequential samples)

to cause differences in parsing results as observed for the sequential samples. It also shows that perplexity can only be used to predict parseability for samples which are not homogeneous. For structurally similar text (as in the randomised samples) perplexity is no reliable measure to forecast parser output quality (note that, while the averaged perplexity for the randomised POS trigram models was identical or even higher than for the sequential models, the variance between the samples was much lower for the randomised samples. This means that homogeneity should not be defined by the overall perplexity in all samples, but by the variance between perplexity for the training and test sets). For measuring parseability for homogeneous text more refined methods are needed, such as the one proposed by Corazza et al. (2008).

5.3 Annotating the TüBa-D/Z in the TiGer Annotation Scheme

In Section 5.2 I showed that comparing treebanks is by no means an easy and straightforward task, and that a fair and unbiased automatic comparison of different encoding schemes is made even more complicated by the fact that other variables, like the actual text in the corpora or sampling methods, might have an impact on results. In order to conduct a meaningful comparison of the impact of different annotation schemes on PCFG parsing, I created a small parallel corpus, containing the same text annotated in the two encoding schemes. This should enable us to abstract away from problems caused by domain variation and text variation.

I extracted a test set of 100 trees from the TüBa-D/Z treebank and manually annotated it following the guidelines in the TiGer annotation manual. Due to the high expenditure of time needed for manual annotation I was able to create a small test set only. To make up for the restricted size I carefully selected the test set by subdividing each of the 44 samples from the TüBa-D/Z treebank into five subsamples with 100 sentences each, and picked the subsample with a sentence length and perplexity closest to the mean sentence length (17.24, mean: 17.27) and mean perplexity computed for the whole treebank (9.44, mean: 9.43). This assures that the test set, despite its limited size, is maximally representative of the treebank as a whole.

I then extracted a training set from the 44 TüBa-D/Z samples (excluding the sentences in the test set). From the TiGer treebank I selected the same number of trees (21898) from the samples 1-44 as well as the first 21898 trees from the samples 45-88 in sequential order and trained the parser on all three training sets (TüBa-D/Z, TiGer1, TiGer2). Then I parsed the test set with the resulting grammars, evaluating the TiGer-trained parser output against the manually created TiGer-style gold-standard of the original TüBa-D/Z strings and the TüBa-D/Z trained parser output for the same strings against the original TüBa-D/Z trees for those strings. Table 5.5 shows the parsing results measured with EVALB and LA.

	TiGer1	**TiGer2**	**TüBa-D/Z**
EVALB	69.84	71.21	83.35
LA	84.91	86.04	88.94

Table 5.5: EVALB and LA results for the manually annotated test set (100 sentences)

As predicted by sentence length and perplexity the LA results for the test set parsed with the TüBa-D/Z grammar is close to the average LA result for the whole TüBa-D/Z (88.95 vs. 88.94; see Table 5.3). For the TiGer grammars parsing TüBa-D/Z-based test strings, however, LA performance drops from 88.36 to 84.91 (TiGer1) and from 88.45 to 86.04 (TiGer2). The better results for TiGer2 imply that the TüBa-D/Z-based test set is more similar to the TiGer2 training set,

an assumption which is supported by the higher word-based perplexity for TiGer2 compared to TiGer1 (643 vs. 599; TüBa-D/Z: 665), and by the average sentence length for the training sets (TiGer1: 17.96, TiGer2: 17.15, TüBa-D/Z: 17.24). However, due to the small size of the test set we cannot make a strong claim. In Section 5.2.1 I showed that, despite coming from the same general domain (newspaper articles, but from two different newspapers), TiGer and TüBa-D/Z are crucially different with regard to the distribution of POS tags, vocabulary size and perplexity. Therefore it is not surprising that the parser trained on a TiGer training set shows lower performance for sentences derived from the TüBa-D/Z. In fact, the results indicate an instance of domain variation, where a parser trained on a data set shows sub-optimal performance when tested on another data set, with properties different from the training set.

5.3.1 Qualitative Evaluation of TiGer and TüBa-D/Z Parser Output

The existence of a small parallel corpus annotated in the TiGer and the TüBa-D/Z annotation schemes allows us to directly compare parser performance for both treebanks. However in addition to the limited size, the differences in categorial and functional labels used in the two annotation schemes often does not support a direct automatic comparison. Here I focus on the grammatical functions describing similar phenomena in both treebanks. Using the same sentences annotated either in the TiGer or the TüBa-D/Z annotation scheme allows us to assess which functions can be compared. Table 5.6 gives an overview over some features of the test set in the TiGer annotation scheme and in the TüBa-D/Z annotation scheme.

	Categorial nodes				*Functional labels*					
	S	NP	PP	AVP	SB	OA	DA	AG	APP	OP
TiGer	155	286	164	85	138	67	11	32	12	16
TüBa-D/Z	159	636	180	105	140	67	10	0	44	24

Table 5.6: Overview over some categorial/functional features in both test sets

Table 5.6 shows that the flat annotation in TiGer leads to a crucially different number of nodes for noun phrases and adverbial phrases for the same sentences. The mismatch in the number of PPs is due to the different annotation of pronominal adverbs, which in TüBa-D/Z are always governed by a PP node, while in TiGer only around one-third of the pronominal adverbs project a PP, the others being either attached to an S or VP node or, less frequently, to an NP, AP or AVP.

With regard to functional labels there are also considerable differences. While some of the basic argument functions like subjects (SB), accusative objects (OA) and dative objects (DA) follow an approximately similar distribution, most other grammatical functions are interpreted differently in both annotation schemes. One example is appositions (APP): the TüBa-D/Z annotation guidelines consider an apposition to be an attribute to a noun which has the same case and does not change the meaning of the noun. They do not distinguish between loosely constructed appositions (e.g.: "Angela Merkel, the chancellor") and tightly constructed appositions (e.g.: "the chancellor Angela Merkel") and treat both as appositional constructions (Figure 5.4). Because of the referential identity of the constituents they do not determine the head of an appositional construction but annotate both constituents as an APP (Figure 5.5).

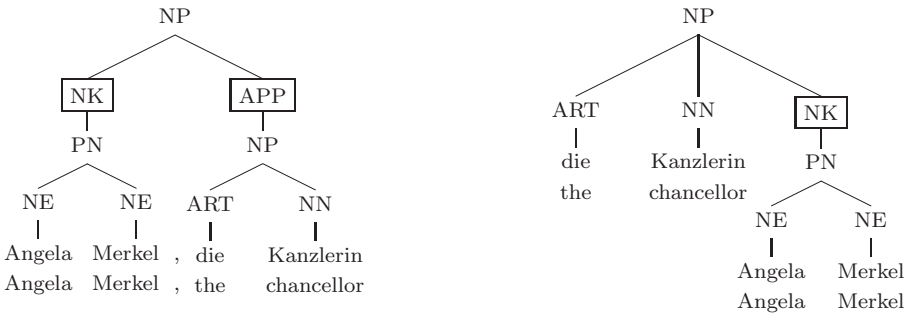


Figure 5.4: The annotation of appositions in TiGer

TiGer only considers loosely constructed appositions which are separated by a comma or another punctuation mark from the preceding element (Figure 5.4). Referential identity is also regarded as a constituting property of an apposition, but in contrast to the TüBa-D/Z the first constituent is annotated as a noun kernel (NK) and the following constituent as an apposition. These differences

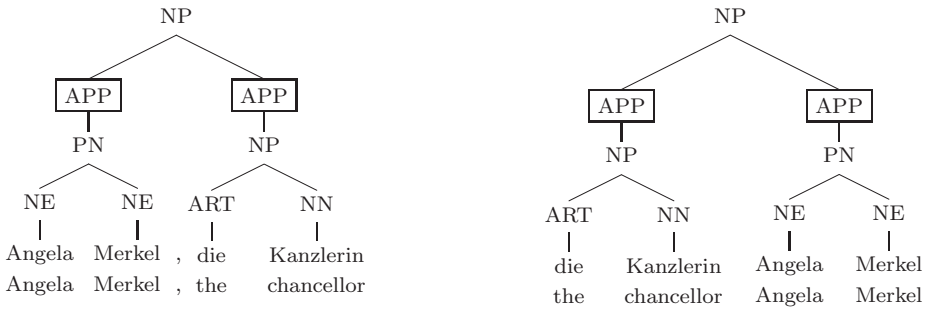


Figure 5.5: The annotation of appositions in TüBa-D/Z

explain the considerable discrepancy in the number of appositions in both test sets.

Another example of the crucial differences in the annotation is postnominal genitives. In TiGer they are annotated with the label AG (Figure 5.6), while the same constituents do not get a label in TüBa-D/Z at all and so are not distinguishable from syntactically similar constructions (Figure 5.7).

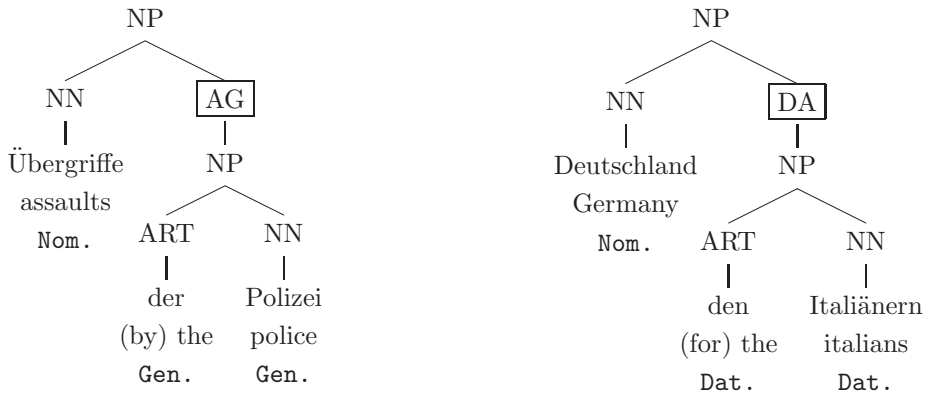


Figure 5.6: The annotation of postnominal genitive and dative attributes in TiGer

However, some of the functions do support a direct comparison between both treebanks, for example subjects, accusative objects, dative objects, predicates and conjuncts of coordinations (Table 5.7). The TüBa-D/Z-trained parser shows better performance for subjects and comparable results for accusative objects, conjuncts and predicates, while it fails to identify dative objects. However, even for grammatical functions which are equally distributed in both treebanks a direct comparison is not straightforward. I will illustrate this for the personal pronoun

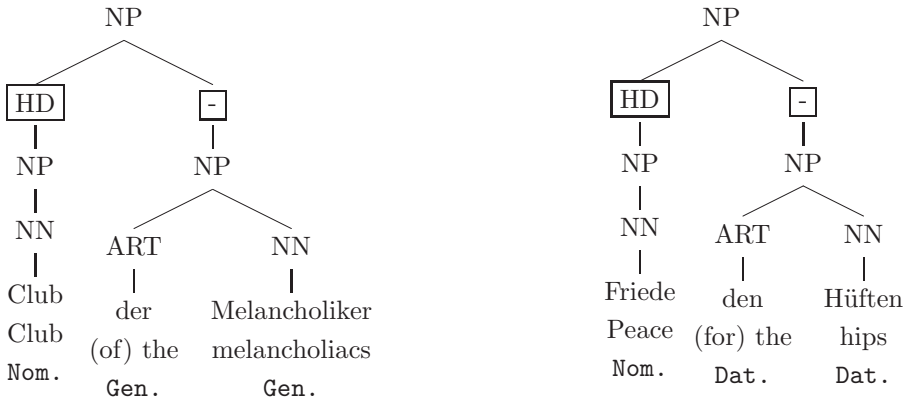


Figure 5.7: The annotation of postnominal genitive and dative attributes in TüBa-D/Z

es (it), which functions either as a subject or as an expletive *es* (it).

	TiGer1			**TiGer2**			**TüBa-D/Z**		
	Prec.	**Recall**	**F-score**	**Prec.**	**Recall**	**F-score**	**Prec.**	**Recall**	**F-score**
subj.	0.64	0.63	0.64	0.66	0.70	0.68	0.73	0.76	0.75
acc. obj.	0.47	0.40	0.43	0.50	0.49	0.50	0.46	0.54	0.50
dat. obj.	0.25	0.18	0.21	0.14	0.09	0.11	0	0	0
conj.	0.47	0.57	0.52	0.44	0.53	0.49	0.53	0.48	0.50
pred.	0.28	0.30	0.29	0.24	0.30	0.27	0.40	0.21	0.28

Table 5.7: Evaluation of functional labels in the test sets

The TüBa-D/Z annotation scheme distinguishes three uses of expletive *es*:

1. Formal subject or object without semantic content
   (e.g. weather verbs)
   (10) Es regnet.
   It rains.
   It's raining.

2. Correlate of an extraposed clausal argument

   (11) Hier bringt es wenig, Bewerbungen herumzuschicken.
   Here brings it little, applications to send around.
   Here it doesn't help to send applications around.

3. Vorfeld-*es* (initial field *es*)

(12) Das bedeutet: Es wird viel schöngeredet, und es passiert nichts.
This means: it is much blandished, and it happens nothing.
This means: there is much blandishing, but nothing happens.

In TüBa-D/Z, formal subjects are annotated as subjects, the correlate *es* is either annotated as a subject modifier or a modifier of an object clause, and the Vorfeld-*es*, which is considered to be a purely structural dummy-element, is assigned the label ES (Table 5.8). The TiGer annotation scheme also distinguishes three uses of the expletive *es*, but annotates them differently. In TiGer *es* as a formal subject is assigned the label EP instead of the subject label. The Vorfeld-*es* as well as the correlate *es* are both annotated as a placeholder (PH).

	formal subject	correlate *es*	Vorfeld-*es*
TIGER	EP	PH	PH
TüBa-D/Z	ON	ON/OS-MOD	ES

Table 5.8: Annotation of expletive *es* (it) in TiGer and TüBa-D/Z

This has major consequences for the test sets, where we have 15 personal pronouns with word form *es*. In the TüBa-D/Z annotation scheme 12 of them are annotated as subjects, the other three as subject modifiers. In TiGer none of them are annotated as a subject. 6 occurrences of *es* are considered to be a placeholder, while the rest are annotated as expletive *es*. If we look at the evaluation results for subjects, 12 of the correctly identified subject relations in the TüBa-D/Z test set are occurrences of expletive *es* (in fact all occurrences of expletive *es* have been assigned the subject label by the parser). The linguistic analysis in the TiGer annotation scheme causes more difficulties for the parser to correctly identify the subject. For the placeholders it has to find the corresponding clause and detect the phrase boundaries correctly, which is more challenging than identifying a single token. Another error frequently made by the TiGer grammar is to mistake an expletive *es* as a subject. Here the TüBa-D/Z grammar has a huge advantage as it annotates formal subjects as regular subjects. Caused by the use of an unlexicalised parsing model in some cases, the TiGer grammar assigns

the label EP to personal pronouns with the word form *er* (he) or *sie* (she). These problems easily explain the gap in evaluation results for subjects between TiGer and TüBa-D/Z and show that even for the same text annotated in the TiGer and in the TüBa-D/Z annotation scheme in Table 5.7, a fair evaluation is not straightforward at all.

5.4 Conclusions

In this Chapter I took a closer look at the two German treebanks, TiGer and TüBa-D/Z, and showed that a fair and unbiased comparison of the different annotation schemes is not straightforward. I showed that, despite coming from the same domain, the content of the two treebanks displays crucial differences with regard to vocabulary and structural homogeneity. The PCA as well as perplexity computed for different models indicate that we may face domain variation problems. In order to assess the impact of different treebank designs on NLP tasks like PCFG parsing, we have to make sure that we exclude these variables from our investigation. Furthermore, sampling methods may influence comparisons.

An attempt to abstract away from these differences resulted in the creation of a small parallel corpus. Even then, differences in linguistic analysis do not allow us to directly compare results automatically and might, in fact, lead to wrong conclusions, as illustrated for the example of expletive *es* (it). In the next chapter we will present a possible way out of the dilemma, using a dependency-based evaluation backed up by a human evaluation of particular grammatical constructions, extracted from the two treebanks.

Chapter 6

TePaCoC - A New Testsuite for Cross-Treebank Comparison

6.1 Introduction

In the last chapter I showed that, due to domain variation problems caused by the actual newspaper articles in the two corpora, and due to differences in linguistic analysis in the two encoding schemes, neither an automatic nor even a manual evaluation of parsing results on a parallel corpus with different annotation schemes is straightforward. Despite all efforts we are still comparing apples with oranges. In this chapter we[13] aim to resolve the puzzle which of the two treebank annotation schemes is more suitable to support data-driven parsing, or at least shed some light on the effect of particular treebank design decisions on the parsing task.

This chapter presents an extensive evaluation of three different parsers, trained on two German treebanks, evaluated with four evaluation measures: the PARSEVAL metric, the Leaf-Ancestor metric, a dependency-based evaluation and a human evaluation of parser performance on a testsuite of particular grammatical constructions, the TePaCoC. The resource (TePaCoC – Testing Parser

[13]This Chapter presents joined work with Sandra Kübler, Wolfgang Maier and Yannick Versley. Sandra and myself created the TePaCoC, developed the error classification system and conducted the human evaluation on the testsuite. I ran the parsing experiments and carried out the PARSEVAL and LA evaluation, while Yannick and Wolfgang carried out the dependency-based evaluation.

Performance on Complex Grammatical Constructions) presented in this chapter takes a different approach to parser evaluation: instead of providing evaluation data in a single annotation scheme, TePaCoC uses comparable sentences and their annotations for 5 selected key grammatical phenomena (with 20 sentences each per phenomena) from both TiGer and TüBa-D/Z resources. This provides a 2 times 100 sentence comparable testsuite which allows us to evaluate TiGer-trained parsers against the TiGer part of TePaCoC, and TüBa-D/Z-trained parsers against the TüBa-D/Z part of TePaCoC for key phenomena, instead of comparing them against a single (and potentially biased) gold standard. To overcome the problem of inconsistency in human evaluation and to bridge the gap between the two different annotation schemes, we provide an extensive error classification, which enables us to compare parser output across the two different treebanks and allows us to trace parser errors back to the underlying treebank design decision. This also gives valuable insights for the future creation of language resources.

Parts of the research presented in this chapter have been published in Kübler et al. (2008) and Kübler et al. (2009).

6.2 Experimental Setup

The limited size of the TePaCoC testsuite (200 sentences) raises suspicions concerning the representativeness of our results. Therefore we also create a larger testset from each treebank with 2000 sentences, in order to complement the human evaluation by an automatic evaluation on a larger data set.

For the experiments, we divided the TüBa-D/Z into a test set with 2000 sentences and a training set, containing the remaining sentences. The 200 sentences in the TePaCoC testsuite were removed from both training and test set. The split was done following the proposal described in Dubey (2004), who split the TiGer treebank into 20 buckets by placing the first sentence of the treebank into bucket 1, the second sentence into bucket 2, and so on. He then combined the content of buckets 1 to 18 into the training set, and used bucket 19 for development and bucket 20 as a test set. As we do not need a development set, we put the last 2000 sentences from buckets 19 and 20 into the test set and use the remaining 25005 sentences for training. For TiGer, we proceed as described for

the TüBa-D/Z (the remaining TiGer sentences beyond the 25005 sentences for the training set were ignored).

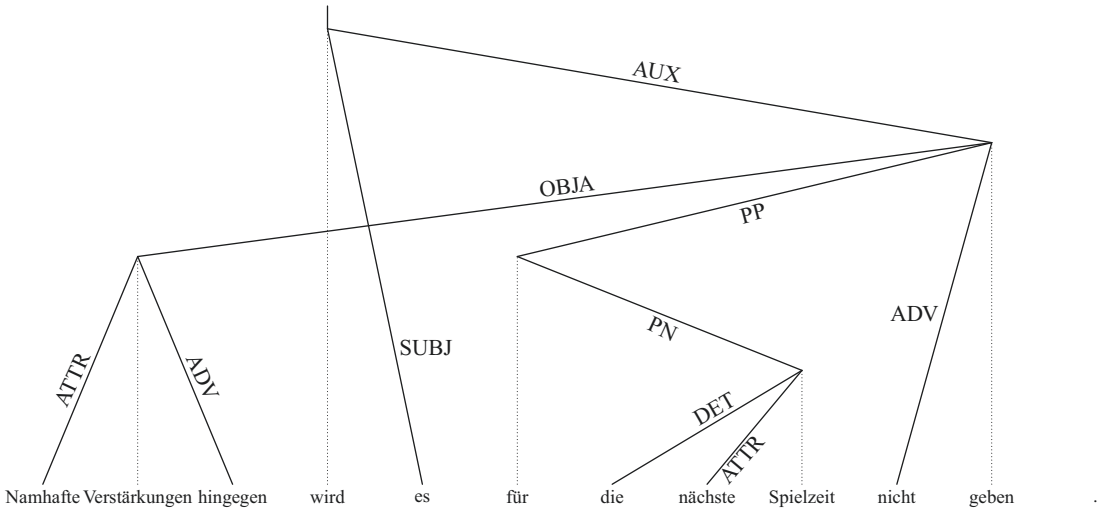
We then trained the unlexicalised parsers BitPar (Schmid, 2004) and LoPar (Schmid, 2000), and the Stanford parser (Klein and Manning, 2003) in its lexicalised and Markovised form[14] on the training set and tested them on the 2000 test sentences as well as on the 200 TePaCoC sentences.

Before extracting the grammars, we resolved the crossing branches in TiGer by attaching the non-head child nodes higher up in the tree and, where grammatical function labels such as *subject* or *accusative object* were directly attached to the terminal node, we inserted an additional unary node to prevent the POS tagset for the TiGer grammar from being blown up artificially, as described in section **??**.

For the dependency-based evaluation, the phrase-structure trees had to be converted into dependencies. We followed the German Dependency Grammar of Foth (2003), who distinguishes 34 different dependency relations. The set of dependencies in the German Dependency Grammar includes five different verb arguments, five types of clausal subordination (infinitive clauses, dependent object clauses, dependent adjunct clauses, full sentences, and relative clauses), and several adjunct relations. Because of inconsistencies between the annotation schemes for TiGer and TüBa-D/Z, we follow Versley (2005) and conflate the labels of prepositional verbal arguments and adjuncts. Foth's dependency grammar annotates exactly one head for each dependent. Figure 6.1 shows an example tree from the TüBa-D/Z treebank, converted to dependencies.

For the TiGer trees we used the dependency converter of Daum et al. (2004), for the TüBa-D/Z trees software by Versley (2005). The conversion process might introduce some noise into the data sets and lower the results, especially when comparing TüBa-D/Z parses with a TiGer gold standard and vice versa. Comparing the accuracy of frequent grammatical functions, however, usually provides a robust estimate for parser output quality.

[14]The parser was trained using the following parameters for Markovisation: `hMarkov=1, vMarkov=2`.



(13) Namhafte Verstärkungen hingegen wird es für die nächste Spielzeit
considerable reinforcements however will it for the next playing time
nicht geben.
not give.
"However, there wont be considerable reinforcements for the next playing season"

Figure 6.1: Dependency tree for a TüBa-D/Z sentence

6.3 TePaCoC - Testing Parser Performance on Complex Grammatical Constructions

Human evaluation is time-consuming and can be applied to small data sets only. Therefore the data has to be chosen carefully. The TePaCoC testsuite contains 200 sentences handpicked from the two German treebanks, TiGer and TüBa-D/Z, with 100 sentences from each. The sentences cover five complex grammatical constructions (20 sentences from each treebank for each construction), which are extremely difficult for a statistical parser to process:

1. PP Attachment: Noun (PPN) vs. Verb Attachment (PPV)
2. Extraposed Relative Clauses (ERC)
3. Forward Conjunction Reduction (FCR)
4. Subject Gap with Finite/Fronted Verbs (SGF)
5. Coordination of Unlike Constituents (CUC)

PP attachment is the canonical case of structural ambiguity and constitutes one of the major problems in (unlexicalised) parsing, since disambiguation often requires lexical rather than structural information (Hindle and Rooth, 1993). The testsuite allows us to investigate which of the different encoding strategies in the two treebanks is more successful in resolving PP attachment ambiguities.

The second construction we included in TePaCoC was extraposed relative clauses, which are a frequent phenomenon in German. According to Gamon et al. (2002), who present a case study in German sentence realisation, 35% of all relative clauses in a corpus of German technical manuals are extraposed, while in a comparable corpus of English technical manuals less than one percent of the relative clauses have been subject to extraposition. This shows that extraposed relative clauses are a frequent phenomenon in German and important to be considered for parser evaluation.

Coordination is a phenomenon which poses a great challenge not only to statistical parsing but also to linguistic theories in general (see for example Sag et al. (1984); Steedman (1985); Kaplan and Maxwell (1988); Pollard and Sag

(1994) for a discussion on different types of coordination in LFG, HPSG, GPSG and CCG respectively).

Harbusch and Kempen (2007) present a corpus study on the TiGer treebank (Release 2), where they investigate cases of clausal coordination with elision. They found 7196 sentences including clausal coordinations, out of which 4046 were subject to elisions. 2545 out of these 4046 sentences proved to be Forward Conjunction Reduction, and 384 sentences contained Subject Gaps with Finite/Fronted Verbs. We included FCR and SGF as two frequent forms of non-constituent coordination in the testsuite. Coordination of unlike constituents is not a very frequent phenomenon and therefore might be considered to be of less interest for data-driven parser evaluation. However, the TiGer treebank (Release 2) contains 384 subtrees with a CUC-labelled constituent, which means that coordination of unlike constituents is as frequent as SGF. Additionally, we choose CUC to be part of the TEPACOC because, from a linguistic point of view, they are quite interesting and put most linguistic theories to the test. There is, of course, a range of phenomena which for linguistic or computational reasons would be of particular interest to be included into the testsuite. Possible examples are equi/raising constructions and verb clusters. For time reasons we did not yet include these, but leave this for future work.

For each of the grammatical phenomena listed above, we selected 20 sentences from TiGer and TüBa-D/Z each with a sentence length ≤ 40.[15] This results in a test set of 200 sentences, 100 from each treebank. Below we describe the different grammatical phenomena and discuss the annotation decisions made in TiGer and TüBa-D/Z for encoding these phenomena.

The differences in treebank design do not support a systematic description of different error types like e.g. span errors, attachment errors or grammatical function label errors, as the same phenomenon might be encoded with the help of GF labels in one treebank and by using attachment in the other treebank. For Extraposed Relative Clauses (ERC), for example, the relation between the extraposed relative clause and the corresponding head noun is expressed through attachment in TiGer, while TüBa-D/Z uses grammatical function labels to encode

[15] We restricted sentence length in the testsuite to $n <= 40$, because many parsers (like the LoPar parser used in our experiments) have considerable problems parsing sentences with a sentence length > 40.

the same relation. In our evaluation, we do not want to count these as different errors but want to generalise over the different representations and evaluate them as the same parser error of not recognising the ERC as a relative clause. Therefore we need well-defined criteria which support a meaningful evaluation and ensure inter-annotator agreement in our human evaluation. We present a descriptive error classification scheme based on empirical data, capturing all potential parser errors on the specific grammatical phenomena.

PP Attachment: Noun (PPN) vs. Verb Attachment (PPV)

The two German treebanks use different strategies to encode prepositional phrases. In TiGer, PPs are annotated as flat tree structures, where the nominal object of the preposition does not project an extra NP, but is directly attached to the PP node. For noun attachment this results in a flat NP in which the PP is attached on the same level as the head noun. For verb attachment the PP is grouped under the VP or the S node (see Example (14) and Figure 1 in the Appendix). In case of attachment ambiguities, TiGer always chooses high attachment. Different edge labels specify the grammatical function of the PP. TiGer distinguishes prepositional objects (OP), postnominal modifiers (MNR), genitive attributes (PG) and verb modifiers (MO). PPs can also be part of a collocational verb construction (CVC), where it is not the preposition, but the noun inside the PP which carries the semantic meaning.

(14) Auf dem Umweg **über die 129a-Ermittlungen** könnten die Bemühungen
By the detour via the 129a-investigations could the efforts
der Autonomen um ein bißchen bürgerliche Respektierlichkeit
of the autonomous activists for a little middle-class respectability
im Keim erstickt werden.
in the bud nipped be.
"With the 129a investigations, the efforts of the autonomous activists for a little middle-class respectability could be nipped in the bud."

The TüBa-D/Z uses more hierarchical structures for the annotation of PPs. For noun attachment the head noun is grouped inside an NP node, with the postmodifier PP as a sister node. Both, the NP and the PP, are then attached to another NP node. For verb attachment the PP is directly attached to the governing topological field. Information about Noun vs. Verb Attachment is expressed

through the use of grammatical function labels in combination with attachment. The TüBa-D/Z distinguishes prepositional objects (OPP), optional prepositional objects (FOPP), unambiguous verbal modifiers (V-MOD), and ambiguous verbal modifiers (MOD). NP postmodifiers get the default label "-" (non-head) (Example (15), Figure 2 (Appendix)).

(15) Wie kann einer sich derart empören über den Wortbruch **bei**
How can one *refl.* so revolt about the breach of promise concerning
den Großflächen-Plakaten, dessen Partei selbst Großflächen-Plakate **in**
the large-scale posters, whose party itself large-scale posters in
Auftrag gegeben und geklebt hat?
commission given and posted has?
"How can someone bristle at the breach of promise concerning the large-scale posters when his party has commissioned and posted such posters?"

Error Classification (PPN vs. PPV)

We consider a PP to be parsed correctly if

1. the PP is recognized correctly;
2. the PP is attached correctly;
3. the PP is assigned the correct grammatical function label.

In TüBa-D/Z, extraposed PPs that are extracted from a preceding NP are not attached directly to the NP, their attachment is shown in the function label. For an extraposed PP in the TüBa-D/Z, incorrect attachment means that the parser assigned a wrong grammatical function label. In such cases, the error code D must be used (Table 6.1).

Error description		TiGer / TüBa
A	correct GF & correct head of PP, span incorrect	
B	correct span, incorrect GF	
C	incorrect span, incorrect GF	
D	wrong attachment	

Table 6.1: Error classification for PP attachment

6.3.1 Extraposed Relative Clauses (ERC)

Extraposed relative clauses in German are treated as adjuncts to the head noun they modify, but there is no agreement in the literature whether they are base-generated locally (Haider, 1996) or whether they obtain their final position through movement (Müller, 2006). In TiGer, relative clauses are attached to the mother node of the head noun, which results in crossing branches for extraposed clauses (Example (16), Figure 3 (Appendix)). The relative clause has the categorial node label S and carries the grammatical function label RC. The relative pronoun is attached directly to the S node.

(16) ...da immer mehr Versicherte nur noch eine **Rente** erhielten, **die**
...that always more insurants just still a pension would receive, which
niedriger ist als die Sozialhilfe
lower is than the social welfare
"... that more and more insured receive a pension lower than social welfare"

In TüBa-D/Z, the extraposed relative clause is located in the final field (NF) and is associated with the node label R-SIMPX. The grammatical function label references the head noun modified by the relative clause (Example (17), Figure 4 (Appendix)). The relative pronoun is embedded inside an NP (NX) which is attached to a C node (complementiser for verb-final sentences).

(17) Warum also soll man homosexuellen Paaren nicht **das** gönnen, **was sie**
Why so shall one homosexual couples not that grant, which they
nun mal für ihr Glück wichtig finden?
now for their luck important find?
"So why shouldn't homosexual couples be granted what they think is important to their happiness?"

In TiGer, the crossing branches make the representation of ERCs more intuitive by encoding the surface word order as well as the deeper dependency relations in a transparent way. After resolving the crossing branches during preprocessing to generate training resources for data-driven parsers following Kübler (2005), this is no longer the case. The relative clause is no longer a sister node of the head noun it modifies, but a sister node of the whole NP. This means that in most cases the dependency between the noun and the relative clause is still recoverable.

Error Classification (ERC)

We consider an ERC to be correct if

1. the clause has been identified by the parser as a relative clause;
2. the clause is associated with the correct head noun;
3. the phrase boundaries have been recognized correctly.

Due to differences in annotation, we have to adapt the error analysis to the two annotation schemes. Table 6.2 shows our error classification for ERC with an error specification for each treebank.

Error description		TiGer	TüBa
(A)	Clause not recognized as relative clause	Grammatical function incorrect	SIMPX label instead of R-SIMPX
(B)	Head noun incorrect	Attachment error	Grammatical function incorrect
(C)	Clause not recognized	Clause not recognized	Clause not recognized
(D)	Sentence boundaries incorrect	Span error	Span error

Table 6.2: Error classification for extraposed relative clauses

In TiGer, the grammatical function label carries the information that the clause is a relative clause. In TüBa-D/Z, the same information is encoded in the categorial node label (R-SIMPX). Therefore, (A) corresponds to a function label error in TiGer and to a categorial node label error in TüBa-D/Z. The relationship between the relative clause and its head noun is expressed through attachment in TiGer and by the use of a grammatical function label in TüBa-D/Z. According to this, (B) is caused by an incorrect attachment decision in TiGer and by a grammatical function label error in TüBa-D/Z. For (C), the parser failed to identify the relative clause at all. In TüBa-D/Z, this is usually caused by a POS tagging error, where the parser failed to assign the correct POS tag to the relative pronoun. In TiGer, error (C) might also be caused by a POS tag error, but there are also cases where the parser annotated the ERC as part of

a coordinated sentence. (D) applies to both annotation schemes: here, the main components of the clause have been identified correctly but the phrase boundaries are slightly wrong.

6.3.2 Forward Conjunction Reduction (FCR)

Forward Conjunction Reduction is a form of non-constituent coordination, in which both conjuncts include an overt head verb. The conjuncts can share the left peripheral context, but there are some restrictions on what else can be shared: only major constituents can be borrowed by the second conjunct. This makes FCR more restricted than for example Right Node Raising, another form of non-constituent coordination where the coordinated constituents share the right-peripheral context. Right Node Raising, in contrast to FCR, also allows for the coordination of many traditional non-constituents.

In TiGer, FCR is annotated as a coordination of sentences. The left peripheral context and the first conjoined verb phrase are grouped as a clause (S), and the second conjunct is projected to an elliptical clause. Both clauses are then coordinated. The information, that the left peripheral context is not only the subject of the first conjunct, but also of the second one, is encoded via a labelled secondary edge (Example (18), Figure 5 (Appendix)).

(18) Die Schatzmeister der beiden Parteien protestierten dagegen und
The treasurers of the both parties protested against it and
kündigten juristische Schritte an.
announced legal action verb part.
"The treasurers of both parties protested and announced they would take legal action."

In TüBa-D/Z, the coordination combines topological fields rather than sentences (Example (19), Figure 6 (Appendix)). As a consequence of the field model, the left peripheral context constitutes the initial field (VF) and is attached higher up in the tree. Here the fact that the NP *Nationalspieler Bode* is the subject of both finite verbs is more transparent than in the TiGer annotation, where the information is encoded by the use of secondary edges (which are not included in the parsing model). Within the field coordination, each conjunct is a combination of the verbal field (LK or VC) and its arguments (MF).

(19) Nationalspieler Bode klagte erneut über eine alte
Member of the national team Bode complained again about an old
Oberschenkelzerrung und konnte nicht das komplette Trainingsprogramm
strain of the thigh and could not the complete training regime
absolvieren.
finish.
"International player Bode again complained about a strain of the femural muscle and could not finish the training."

Error Classification (FCR)

We consider an FCR to be parsed correctly if

1. the parser has identified the coordination;
2. the parser has assigned the subject label to the right node;
3. no other node in the first or second constituent has been associated with the subject label.

Here, with the exception of span errors, the annotation schemes allow us to use the same error specification for both treebanks (Table 6.3).

	Error description	TiGer / TüBa
A	Parser incorrectly annotates subject in one of the constituents	
B	Parser fails to identify subject	
C	Coordination not recognized	
D	Second subject in first conjunct	
E	Span error	(only in TüBa-D/Z)

Table 6.3: Error classification for forward conjunction reduction

6.3.3 Subject Gap with Fronted/Finite Verbs (SGF)

In SGF constructions the shared constituent is not embedded in the left peripheral context, as it is the case for FCR, but in the middle field of the first conjunct. This poses a challenge for theoretical linguistics, where SGF has been analysed

as an asymmetric form of coordination (Wunderlich, 1988; Höhle, 1990) as well as a symmetric coordinated construction (Steedman, 1990; Kathol, 1999). Both approaches bear their own problems. The phrase-structure-based approaches of Höhle (1990) and Heycock and Kroch (1993) lead to extraction asymmetries and violate constraints like the Across-the-Board (ATB) extraction constraint. In contrast, Steedman (1990) analyses SGF as a form of gapping, which is criticised by Kathol (1999). Kathol argues that only subjects can be extracted from the middle field and points out that Steedman's analysis does not predict the ungrammaticality of object gaps, as shown in Example (20).

(20) Statt dessen leugnet **man** Tatsachen und verdreht sie.
Instead denies one facts and twists them.
"Instead, the facts are denied and twisted."

Kathol (1999) presents a linearisation-based approach which relies on the topological field model. In his analysis Kathol separates constituent relations from word order and establishes structural and functional constraints which allow him to capture word order asymmetries in SGF constructions. Frank (2002), however, states that the constraints used in Kathol's analysis are not well motivated. Instead, Frank (2002) proposes an LFG-based analysis which combines symmetric and asymmetric approaches. She presents a solution in which SGF is analysed as a symmetric coordination in c-structure, where the subject, which is embedded inside the first constituent and so inaccessible for the second constituent, is bound by asymmetric projection of a *grammaticalised discourse function* (GDF) on the level of f-structure.

None of the linguistic analyses described above can be associated directly with one of the annotation schemes of the two German treebanks. However, the TüBa-D/Z with its layer of topological fields seems to be closer to theories like the one of Kathol (1999), while the TiGer treebank, which is partly based on a hand-corrected version of the output of the German ParGram LFG grammar, should be more suitable to represent theories like the one of Frank (2002). Therefore it will be interesting to see the differences in performance of parsers trained on the two treebank annotation schemes on non-constituent coordinations, especially on SGFs.

In TiGer, SGFs are encoded as a coordination of sentences (CS) (Example (20), Figure 7 (Appendix)). The subject is realised in the first constituent and

can be identified by the grammatical function label SB (subject). With the help of labeled secondary edges (SB), TiGer makes explicit that the subject of the first constituent should also be interpreted as the subject of the second constituent. In TüBa-D/Z, SGFs are treated as a complex coordination of fields (FKOORD) (Example (21), Figure 8 (Appendix)). As in TiGer, the subject is part of the first constituent, where it is attached to the middle field and has the functional label ON (nominative object). Both constituents are associated with the functional label FKONJ (conjunct with more than one field).

(21) Immer kommt **einer** und stiehlt mir meine Krise.
Always comes someone and steals me my crisis.
"Every time, someone comes and steals my crisis."

Error Classification (SGF)

We consider an SGF to be parsed correctly if

1. the parser has identified the coordination;
2. the parser has assigned the subject label to the right node in the first constituent;
3. no other node in the first or second constituent has been associated with the subject label.

Here, the annotation schemes allow us to use the same error specification for both treebanks (Table 6.4).

	Error description	TiGer / TüBa
A	Parser incorrectly annotates subject in second conjunct	
B	Parser fails to identify subject in first conjunct	
C	Coordination not recognized	
D	Parser annotates additional subject in first conjunct	
E	Parser fails to identify the verb in the sentence	

Table 6.4: Error classification for subject gap with fronted/final verb

6.3.4 Coordination of Unlike Constituents (CUC)

The sentences in TePaCoC cover three types of coordinations of unlike constituents: VPs coordinated with adjectival phrases (AP), VPs coordinated with NPs, and clauses (S) coordinated with NPs. Here, we will concentrate on the second type (VP-NP), which shows the greatest differences between the two annotation schemes. In TiGer, the coordination is rather straightforward: the VP and the NP project to a coordinated phrase (CO). The functional labels for the conjuncts (CJ) describe their conjunct status, while the coordination gets the functional label of the verb phrase (OC). The grammatical function of the NP remains unspecified (Example (22), Figure 9 (Appendix)).

(22) Das ist eigentlich ein Witz und nicht zu verstehen.
This is actually a joke and not to understand.
"This actually is a joke and hard to understand."

In the TüBa-D/Z, CUCs are annotated as a coordination of complex topological fields. The VP is represented as a combination of the verbal field and the middle field (MF). The NP in the first conjunct is projected to the MF, before both conjuncts are coordinated. Here, the grammatical functions are retained in the constituents under the MFs (Example (23), Figure 10 (Appendix)).

(23) Die Älteren sind teurer, haben familiäre Verpflichtungen und oft
The elderly are more expensive, have familial commitments and often
ein Haus abzuzahlen.
a house to repay.
"The elderly are more expensive, have family commitments and often have to pay off a house."

Error Classification (CUC)

Since the two annotation schemes differ drastically in the annotation of coordinations of unlike constituents, we decided to use a correct/incorrect distinction only. A CUC is considered correct if

1. the constituents are recognized with correct spans;
2. the parser recognised the heads of all constituents correctly.

6.4 Constituent Evaluation

Table 6.5 shows constituent-based evaluation results for the 2000 sentence test-sets, measured with EVALB and LA . As discussed in Chapter 3, there is a wide gap between EVALB results for the TiGer and the TüBa-D/Z model, while LA scores for both treebanks are much closer. This is due to the fact that EVALB has a strong bias towards annotation schemes with a high ratio of nonterminal vs. terminal nodes as in the TüBa-D/Z (see Section 4.3.5). Additionally, there is a clear improvement from BitPar to LoPar to the Stanford parser for both treebanks, which is consistent for both constituency-based evaluation metrics. The differences between BitPar and LoPar are rather surprising since both parsers are based on the same principles. The difference may be due to the internal translation of the grammar into CNF in BitPar (Schmid, 2004), or to differences in smoothing. The Stanford parser obviously profits from the combination of lexicalisation and Markovisation.

Table 6.6 shows evaluation results for the TEPACOC sentences. Compared to our 2000 sentence test sets, most EVALB and LA scores are considerably lower. This confirms our intuition that the TEPACOC sample constitutes a challenge for statistical parsers. Again, we observe the same parser ranking as for the larger test sets, and again the TüBa-D/Z results are higher than the ones for TiGer. This shows that, apart from being more difficult to parse, the sentences in TEPACOC show the same properties as the larger test sets.

6.5 Dependency Evaluation

The bias of both constituent-based evaluation measures (cf. Section 4.3.5) does not support a cross-treebank comparison of the results. Therefore we resort to

	TiGer			**TüBa-D/Z**		
	Bit	**Lop**	**Stan**	**Bit**	**Lop**	**Stan**
EVALB	74.0	75.2	77.3	83.4	84.6	88.5
LA	90.9	91.3	92.4	91.5	91.8	93.6

Table 6.5: EVALB and LA scores (2000 sentences)

		TiGer			TüBa-D/Z		
		Bit	**Lop**	**Stan**	**Bit**	**Lop**	**Stan**
EVALB	**ERC**	71.7	73.0	76.1	80.6	82.8	82.8
	FCR	76.6	77.7	81.3	84.0	85.2	86.7
	PPN	71.2	73.9	83.6	86.2	87.4	89.2
	PPV	71.9	76.5	78.7	84.3	85.0	91.9
	CUC	55.9	56.5	63.4	78.4	73.6	76.6
	SGF	73.3	74.1	78.6	73.6	76.6	78.4
	ALL	**69.64**	**71.07**	**75.82**	**81.20**	**83.51**	**84.86**
LA	**ERC**	85.3	86.1	84.8	89.3	89.8	91.0
	FCR	91.2	89.0	91.0	92.0	93.4	88.7
	PPN	87.1	88.7	91.0	94.2	94.3	94.4
	PPV	88.4	88.9	86.4	91.3	90.5	94.7
	CUC	78.0	78.4	78.3	82.2	85.5	84.9
	SGF	89.1	89.7	87.5	90.9	94.4	88.5
	ALL	**86.26**	**86.42**	**86.09**	**89.42**	**91.13**	**89.84**

Table 6.6: EVALB (labeled) bracketing and LA scores (TEPACOC)

a dependency-based evaluation (Lin, 1995, 1998; Kübler and Telljohann, 2002), which is considered to be more neutral with regard to the underlying annotation scheme. Arguably, the results of a dependency-based evaluation give a more meaningful insight into parser errors than the number of correctly matched brackets in the tree. Another great advantage of the dependency-based evaluation concerns the resolving of crossing branches in TiGer. The constituency-based evaluation measures can only be applied to trees with crossing branches resolved. This means that, for TiGer, we evaluate against a lossy representation, which certainly distorts results. By contrast, the dependency-based evaluation allows us to evaluate parser output against the original treebank trees including non-local information.

Table 6.7 shows the results for the dependency evaluation of the 2000 sentence test sets. We observe the same parser ranking as in the constituent-based evaluation, and again this is consistent for both treebanks. For unlabelled accuracy scores (UAS), the Stanford parser trained on the TüBa-D/Z gives the best

	TiGer			TüBa-D/Z		
	Bit	**Lop**	**Stan**	**Bit**	**Lop**	**Stan**
LAS	78.8	80.5	81.6	71.3	72.8	75.9
UAS	83.0	84.5	85.6	81.7	83.4	86.8

Table 6.7: Labeled/unlabeled dependency accuracy for the 2000 test sentences

	TiGer			TüBa-D/Z		
	Bit	**Lop**	**Stan**	**Bit**	**Lop**	**Stan**
SUBJ	80.2	81.1	78.7	74.6	75.3	76.1
OBJA	55.6	58.4	59.5	42.4	45.8	52.9
OBJD	11.6	11.5	14.1	12.9	13.3	13.1
PP	71.1	72.2	78.2	68.1	69.1	75.6
CL-SUB	57.0	58.2	60.9	45.8	47.5	52.1

Table 6.8: Dependency F-measure for the 2000 test sentences: nominal verb arguments (subjects and accusative/dative objects), PP attachment and clause subordination (including infinitive and relative clauses as well as adjunct and argument subordinated clauses and argument full clauses)

results, but for labelled accuracy the results for all TiGer-trained parsers are far better than for the same parsers trained on the TüBa-D/Z. This result clearly contradicts the constituent-based evaluation.

Table 6.8 gives dependency F-scores for specific dependency relations. The results are mostly consistent with the accuracy scores in Table 6.7, showing better LAS results for the TiGer-trained parsers and replicating the parser ranking Bitpar < LoPar < Stanford. For subjects, however, the TiGer-trained Stanford parser shows a lower performance than the two unlexicalised parsers, and also for dative objects the ranking is slightly distorted with BitPar outperforming the TiGer-trained LoPar parser. For PP attachment the Stanford parser gives by far the best results, which is not surprising, as the disambiguation of PP attachment is dependent on lexical information.

The accuracy scores for the TePaCoC testsuite paint the same picture as the results for the 2000 sentences test sets. For the TiGer-trained parsers we achieve lower unlabelled dependency accuracy, but far better results for labelled accu-

		TiGer			TüBa-D/Z		
		Bit	**Lop**	**Stan**	**Bit**	**Lop**	**Stan**
LAS	**ERC**	76.2	76.0	77.4	71.6	71.8	71.1
	FCR	79.5	74.4	81.8	78.5	81.0	79.3
	PPN	76.8	79.7	87.0	75.5	76.1	76.1
	PPV	73.6	80.9	79.2	65.8	67.9	71.5
	CUC	65.2	67.0	70.7	57.5	63.0	60.9
	SGF	76.1	77.2	79.3	74.0	77.7	75.1
	ALL	**73.3**	**73.9**	**76.8**	**69.3**	**72.7**	**70.3**
UAS	**ERC**	81.1	80.8	82.0	79.1	80.5	79.1
	FCR	82.7	77.8	85.6	85.4	88.2	88.7
	PPN	84.2	86.4	89.3	84.8	85.3	85.9
	PPV	78.1	86.0	86.0	81.3	82.9	88.6
	CUC	69.7	71.5	74.7	66.1	72.0	73.6
	SGF	81.7	82.5	83.6	82.8	86.2	85.4
	ALL	**78.1**	**78.7**	**81.0**	**78.3**	**81.9**	**81.7**

Table 6.9: Labeled/unlabeled dependency accuracy for the TePaCoC testsuite

racy compared to the TüBa-D/Z-trained parsers. Table 6.9 lists the LAS/UAS for the whole testsuite as well as for the particular constructions. The scores for specific phenomena, however, are not really significant because of the small number of sentences (20 sentences for each phenomenon; PPN and PPV count as one phenomenon). We should also keep in mind that the dependency evaluation does not solely focus on the particular grammatical construction, but evaluates all dependency relations in the trees. For the TiGer-trained sentences we obtain the same parser ranking as before (BitPar < LoPar < Stanford), for the TüBa-D/Z the Stanford results are lower than the results for LoPar. While for PP verb attachment in the TüBa-D/Z parsing model the lexicalised Stanford parser is superior to the unlexicalised parsers, lexicalisation does not help to parse the different types of coordination in the testsuite. Especially for CUC and SGF, results for the Stanford parser are significantly lower than for LoPar. A possible explanation might be that the additional layer of topological fields prevents the benefits of lexicalisation on clause level.

6.6 Manual Evaluation of TePaCoC Phenomena

The results for the dependency evaluation clearly contradict the constituent-based evaluation using EVALB and LA. In Chapter 3 I showed that the constituent-based measures are highly sensitive to the data structures in the treebanks. Therefore we believe that the dependency-based evaluation gives a more meaningful assessment of the quality in the parser output. To back up our claim we add a human evaluation of the testsuite. Here we are interested in how the parsers perform on handling particular grammatical constructions, as included in the TePaCoC testsuite. This allows us to concentrate on our core phenomena (rather than the cumulative scores over all dependencies in the sentences in Table 6.9).

Table 6.10 shows the results for a human evaluation for the different phenomena in TePaCoC. The rightmost column gives the number of occurrences of the particular phenomenon in the testsuite. To keep things simple we do not list the different error categories but rather the total number of correctly parsed constructions in TiGer and TüBa-D/Z. For extraposed relative clauses (ERC) and for both types of asymmetric coordinations (FCR, SGF), we observe distinctly better results for the TiGer-trained parsers. For relative clauses, in TiGer the relative pronoun is directly attached to the relative clause, which makes it easier for the parser to recognise the whole clause as a relative clause. Another advantage is due to our method of resolving crossing branches in TiGer. Due to the conversion the relative clause, which originally was attached to the NP node of the head noun, is now a sister node of the NP and attached to the VP or S mother node of the NP. This again makes it easier for the TiGer-trained parsers to process extraposed relative clauses correctly, but still enables us to reconstruct the dependency between the head noun and the relative clause in most cases.

For the two non-constituent coordinations, FRC and SGF, the two annotation schemes make different decisions with regard to the level of attachment for the coordination. In TiGer, the coordination is attached at the clause level while TüBa-D/Z coordinates complex fields. This results in a higher number of possible attachment locations in the TüBa-D/Z model and makes it harder for the parser to attach FCR and SGF constructions correctly.

Coordinations of Unlike Constituents (CUC) are extremely difficult to parse

	TiGer			TüBa-D/Z			
	Bit	**Lop**	**Stan**	**Bit**	**Lop**	**Stan**	**Total**
ERC	20	19	19	0	0	3	41
FCR	26	27	23	11	9	13	40
PPN	9	9	16	15	14	14	60
PPV	15	16	18	14	13	18	62
CUC	6	8	5	6	7	5	39
SGF	18	20	20	7	10	8	40

Table 6.10: Correctly parsed constructions in TiGer and TüBa-D/Z (human evaluation)

for both the TiGer- and the TüBa-D/Z-trained parsing models. The unlexicalised parsers yield slightly better results, but the number of CUC sentences is too small to make a strong claim.

For PP Verb Attachment (PPV), the combination of lexicalisation and Markovisation clearly helps: the Stanford parser outperforms both unlexicalised parsers. For PP Noun Attachment (PPN), the lexicalised Stanford parser trained on TiGer outperforms the unlexicalised TiGer-trained parsers and also the results for the Stanford parser trained on the TüBa-D/Z. The unlexicalised parsers do much better when trained on the more hierarchical annotation of the TüBa-D/Z, which apparently makes it easier to disambiguate constituent structure for noun attachment. However, there might be another reason for the better performance of the TüBa-D/Z-trained parsers. The newspaper articles in the two corpora show a very different distribution of noun versus verb attachment: around 74% of all *noun PP* sequences in TüBa-D/Z in fact show noun attachment, while in TiGer only approximately 57% of those PPs are attached to the noun. It is hard to decide if the better results for the TüBa-D/Z-trained parsers are due to the tree structure in the TüBa-D/Z, or if they are just an artefact of the higher ratio of noun attachments in the corpus.

In combination with the dependency-based evaluation, the manual evaluation shows that while EVALB and, to a smaller degree, LA favor the TüBa-D/Z annotation scheme, many of the phenomena covered in TePaCoC are easier to parse with TiGer. Obviously, none of the parsers' models are able to cover

the hierarchical structure of TüBa-D/Z successfully. A solution which immediately comes to mind is the use of parent encoding (Johnson, 1998), a treebank transformation technique which adds local (vertical) context information to the trees. Each node is augmented with the syntactic node label of its parent node (for parent annotation) and with the node label of its grandparent node (for grandparent annotation). In our parsing experiments with the Stanford parser we set the parameter for vertical Markovisation (hence parent annotation) to 2 for both treebanks, which means that the categorial node labels in the trees are augmented with the information about the syntactic node labels of their parent nodes. We run two additional experiments. In the first experiment we set the parameter for vertical Markovisation for the Stanford parser to 1, which means that no parent encoding is used. In the second experiment we set the parameter for vertical Markovisation to 3, which means that the parsing model is enriched with grandparent information for each node in the tree. We parsed the subset of the TePaCoC containing the ERC sentences with the new parameter settings. It is obvious that in order to recognise a clause as a relative clause, the parser heavily relies on the information whether there is a relative pronoun governed by the node. We expected, that for the first experiment results would deteriorate, while for the second experiment results should improve. To our surprise there was no difference between the parser output for `vMarkov=1` and `vMarkov=2`. We observed differences between the parser output for the settings `vMarkov=2` and `vMarkov=3`, but these differences did not concern the recognition of ERC constructions in the test sentences. This means that the problem inherent in the more hierarchical annotation of the TüBa-D/Z annotation scheme cannot be solved easily by techniques like parent or grandparent encoding.

The manual evaluation also backs up the dependency-based evaluation and gives more evidence for the already strong suspicion that the PARSEVAL metric, while being a useful tool to assess parser performance for parsers trained on the same training and test sets, is not adequate to give a linguistically motivated assessment of the quality of parser output across treebanks and languages.

6.7 Conclusions

In this chapter, we showed how human evaluation of a comparable corpus of complex grammatical constructions with 100 sentences from each of the TiGer and TüBa-D/Z treebanks allows us to detect error types and trace them back to the annotation decision underlying the error. Our main findings are: TiGer benefits from the flat annotation which makes it more transparent and straightforward for the parser to detect constructions like Extraposed Relative Clauses, Forward Conjunction Reduction, or Subject Gapping with Fronted/Finite Verbs, while TüBa-D/Z suffers from the more hierarchical structure where relevant clues are embedded too deep in the tree for the parser to make use of it. While the additional layer of topological fields in TüBa-D/Z increases the number of possible attachment positions, it also reduces the number of rules in the grammar and improves the learnability especially for small training sets.

In the next chapter I give a short overview of Lexical Functional Grammar and provide some background on treebank-based automatic acquisition of deep LFG resources.

Chapter 7

Treebank-Based Deep Grammar Acquisition - Background

In the previous chapters I discussed problems arising from cross-treebank comparisons and showed how particular treebank design decisions influence PCFG parsing performance. In the remainder of the thesis I expand the parsing task and test the adequacy of two different treebank annotation schemes as part of an architecture for treebank-based deep grammar acquisition. Chapter 7 provides an overview of data-driven deep grammar acquisition, focussing on the acquisition of LFG resources for English. I review work on multilingual treebank-based grammar acquisition and describe early efforts to port the LFG annotation algorithm to the German TiGer treebank (Cahill et al., 2003; Cahill, 2004; Cahill et al., 2005). Chapter 8 describes my own work on treebank-based grammar acquisition for German. I present a substantially revised, extended and improved method for the acquisition of deep, wide-coverage LFG resources for German, based on the two different treebanks (TiGer and TüBa-D/Z). An extensive evaluation and error analysis sheds some light on the impact of treebank design on the grammar acquisition task.

7.1 Treebank-Based Automatic Acquisition of Deep LFG Resources

Recent years have seen the development of a new and active research area to automatically acquire deep linguistic resources encoding detailed and fine-grained linguistic information from treebanks. The research uses Tree Adjoining Grammar (TAG), Categorial Grammar (CCG), Head-Driven Phrase Structure Grammar (HPSG) and Lexical Functional Grammar (LFG), and, to date, has mostly concentrated on English.

Hockenmaier and Steedman (2002a) converted the Penn-II treebank into a CCG-derivation treebank. They carried out an extensive preprocessing of the Penn treebank cleaning up errors and modifying tree structures according to the requirements of the CCG grammar formalism, binarising the trees and converting them into CCG derivations and categories. They added co-indexations to lexical categories to represent long-distance dependencies and generated predicate-argument structures. The resulting CCGBank (Hockenmaier and Steedman, 2005) is based on 99.44% of the original Penn treebank trees. The CCG resources extracted are then used for statistical parsing (Hockenmaier and Steedman, 2002b), employing a standard CKY chart parser and a variety of probability models. Clark and Curran (2003, 2004) extended Hockenmaier and Steedman's work by applying log-linear parsing models to CCG. For large grammars like the CCG grammar, this requires a very large amount of computational resources. Therefore, Clark and Curran (2003), following Miyao and Tsujii (2002), applied the inside-outside algorithm to a packed representation of the parse forest, allowing them to compute the models efficiently.

Nakanishi et al. (2004) and Miyao and Tsujii (2005) developed an approach based on the HPSG framework, which enables them to extract an HPSG lexicon from the Penn-II treebank and to develop and train probabilistic models for parsing. They use discriminative log-linear models for parse disambiguation, working on a packed representation of parse forests.

Cahill et al. (2002, 2003, 2005) and Cahill (2004) developed a method to automatically annotate the Penn-II treebank with LFG F-structures to extract wide-coverage LFG resources. Their work on English provides a method for wide-coverage, deep, constraint-based grammar acquisition, with results (Cahill, 2004;

Cahill et al., 2008) in the same range as or better than the best hand-crafted grammars developed for English (Briscoe and Carroll, 2002; Kaplan et al., 2004). The next section briefly outlines the main concepts of LFG and gives an overview of the core component of the treebank-based LFG acquisition architecture: the LFG F-structure annotation algorithm.

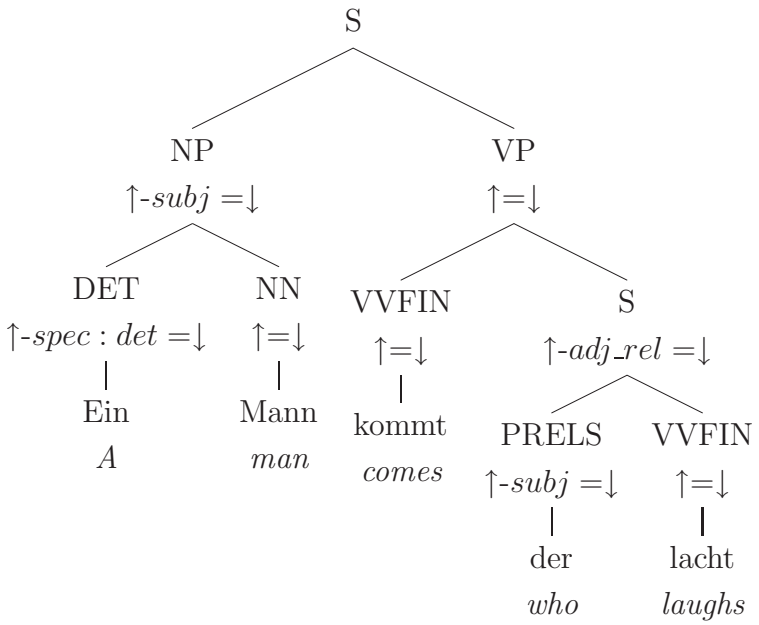
7.1.1 Overview of Lexical Functional Grammar

Lexical Functional Grammar (LFG) (Bresnan, 2000; Dalrymple, 2001) is a constraint-based theory of grammar with at least two levels of representation: Constituent Structure (c-structure), where strings and the hierarchical grouping of phrases are expressed through context-free phrase structure trees, and Functional Structure (F-structure), which represents more abstract linguistic information in the form of grammatical functions (e.g. subject, object, modifier, topic). C-structure is determined by context-free phrase structure rules (1), and functional annotations on c-structure nodes link c-structure categories to their corresponding grammatical functions in F-structure.

$$\begin{array}{llll} (1) & \text{S} & \rightarrow & \begin{array}{ll} \text{NP} & \text{VP} \\ (\uparrow \text{SUBJ})=\downarrow & \uparrow=\downarrow \end{array} \end{array}$$

The grammar rule in (1) states that a sentence (S) can consist of a noun phrase (NP) followed by a verb phrase (VP), and the functional annotations identify the F-structure of the NP as the subject of the sentence ((↑ SUBJ)=↓), while the VP constitutes the head (↑=↓). C-structure representations are the same kind of data structures as the CFG trees in the Penn treebank, but without the traces. F-structures encode more abstract linguistic information approximating to predicate-argument-adjunct structure, dependencies or simple logical forms. Figure 7.1 shows a c-structure tree annotated with LFG F-structure equations together with its corresponding F-structure.[16] The subject of the main clause is also the subject of the extraposed relative clause, which is shown by the arc

[16] Lexical equations are omitted for reasons of clarity.



Ein Mann kommt, der lacht.
A man comes, who laughes

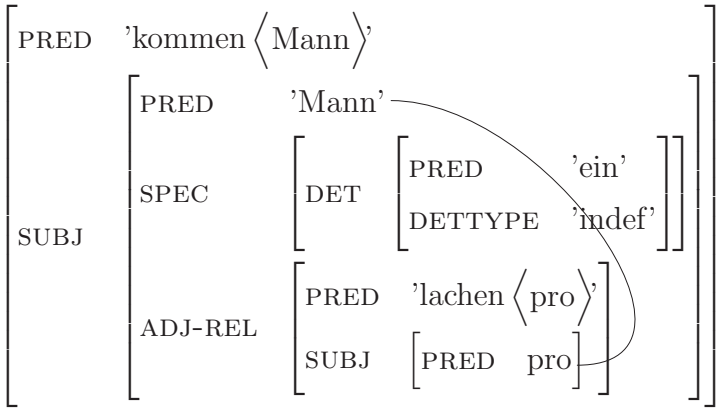


Figure 7.1: LFG c-structure and F-structure

in Figure 7.1, pointing from the subject Mann (*man*) in the main clause to the pronoun (*pro*) which is subject of the embedded relative clause.

LFG is a non-transformational grammar theory where syntactic phenomena are treated through the specification of rules and constraints in the lexicon. Similar to HPSG and CCG, the lexicon plays an important role in LFG.

7.1.2 Automatic F-structure Annotation of the English Penn-II Treebank

In order to automatically add F-structure information to the Penn treebank, Cahill et al. (2002) and Cahill (2004) exploit information encoded in the original treebank. The Penn treebank provides categorial information (like NP or PP) and additional functional tags such as logical subject, surface subject, predicate etc. Long-distance dependencies are expressed in terms of traces and co-indexation in CFG trees. Unlike in the CCG and HPSG-based approaches, in the LFG-based approach the Penn-II treebank trees are not cleaned-up or restructured into different trees. The phrase structure trees remain as they are, while a further level of annotation is added by an F-structure annotation algorithm: functional equations describing F-structures.

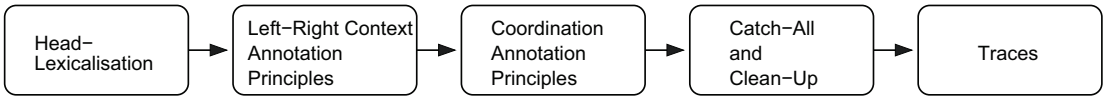


Figure 7.2: Architecture of the F-structure annotation algorithm

The F-structure annotation algorithm is designed in a modular way (Figure 7.2). The first step in the annotation process is the head-lexicalisation module. This procedure is based on the head-finding rules of Magerman (1995), which have been slightly modified. After the head and mother categories for each phrase have been determined, left-right context annotation principles exploiting configurational properties of English are applied to assign functional annotations to each phrasal category. The annotation principles are based on hand-crafted Left-Right Annotation Matrices, which, for each phrasal category are based on the most frequent CFG rules expanding this node. This results in high coverage but in some cases may lead to overgeneralisations. These incorrect annotations (exceptions) have to be detected and corrected in a later Catch-All and Clean-

Up stage during the annotation process. Before that, however, the Coordination Annotation Principles are applied to trees, dealing with different kinds of coordinations. This task has been assigned to a designated module in order to keep the Left-Right Annotation Principles simple and perspicuous. After the Catch-All and Clean-Up module has finished, the trees have been annotated with basic functional annotations, but long-distance dependencies are still unsolved. The F-structures defined by these preliminary annotations are referred to as "proto F-structures". In order to get "proper" F-structures, where long-distance dependencies are resolved, the annotation algorithm provides the Traces module, which exploits the information provided by the traces and co-indexation in the Penn-II treebank and represents long-distance dependencies as corresponding reentrancies in F-structure.

7.1.3 Using F-structure Information to Guide Parsing

To date most probabilistic treebank-trained parsers are not able to produce traces and co-indexation in CFG output trees, as present in the original Penn-II treebank. Without traces and co-indexation the F-structure Annotation Algorithm is only able to produce proto F-structures with long-distance dependencies unsolved. Cahill et al. (2004) present a solution to this problem: for parsing they resolve LDDs on the level of F-structures. Their method is based on finite approximations of LFG functional uncertainty equations (Kaplan and Zaenen, 1988; Dalrymple, 2001), and subcategorisation frames (O'Donovan et al., 2004) automatically learned from the F-structures generated for the Penn-II treebank.

Cahill (2004) and Cahill et al. (2004) developed two parsing architectures: the Pipeline Model and the Integrated Model (Figure 7.3). In the Pipeline Model a PCFG or a history-based, lexicalised generative parser is extracted from the training sections 01-22 of the original unannotated Penn-II treebank. The parser is used to parse raw text into CFG trees. The parser output is handed over to the annotation algorithm, where all the nodes in the parse tree are annotated with LFG functional equations. The F-structure equations are then handed over to a constraint solver, which generates F-structures.

In the Integrated Model the original treebank trees are first automatically annotated with F-structure equations. Then a PCFG is extracted from the anno-

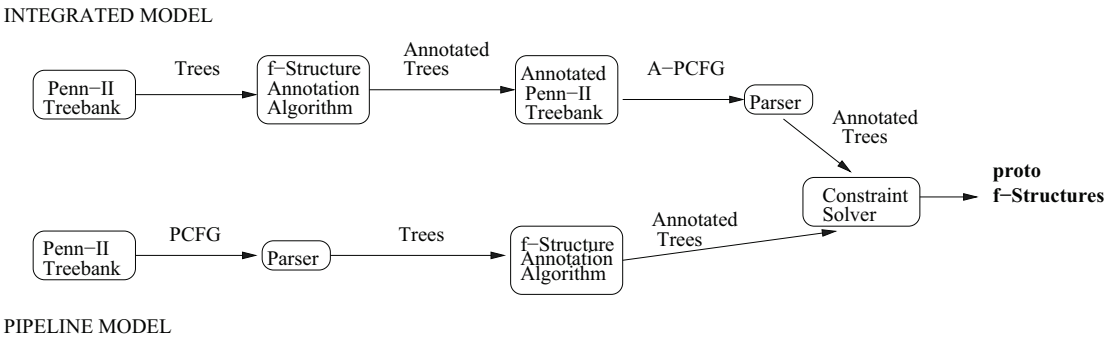


Figure 7.3: Two parsing architectures for English

tated trees. The annotated PCFG is then used to parse new text, which outputs a parse tree complete with functional equations. These equations again are collected and passed over to the constraint solver, which generates the F-structures. At this stage both models have parsed raw text into proto F-structures, where LDDs remain unsolved.

7.1.4 Extracting Subcategorisation Frames from the F-structures Generated from the Penn-II Treebank

The F-structure-annotated Penn-II treebank offers rich semantic information in terms of predicate-argument structure and can therefore be used for the extraction of subcategorisation frames (semantic forms). Van Genabith et al. (1999) and O'Donovan et al. (2004, 2005a) developed a method where, based on a set of subcategorisable grammatical functions, for each F-structure and each level of embedding the PRED value on that level is determined and all the subcategorisable grammatical functions present on that level are collected. The semantic forms extracted in this way are then associated with conditional probabilities and can be used for e.g. the resolution of long-distance dependencies in parsing, among others.

7.1.5 Resolving LDDs on F-structure Level for Parser Output

Parsing initially results in proto F-structures (Figure 7.3), derived from F-structure-annotated probabilistic parser output trees, where long-distance dependencies remain unsolved. In LFG long-distance dependencies are resolved with the help of functional uncertainty equations (Kaplan and Zaenen, 1988; Dalrymple, 2001). These uncertainty equations define a path in the F-structure between the surface position of a linguistic element in F-structure and the location where it should be interpreted semantically. Burke et al. (2004a); Cahill et al. (2004); Cahill (2004) show that functional uncertainty paths can be automatically approximated through the extraction of paths for co-indexed material in the automatically F-structure-annotated Penn-II treebank. For extracted paths conditional probabilities are computed. The LDD resolution algorithm takes these finite approximations of functional uncertainty paths and the extracted semantic forms, and, given an LDD trigger (such as FOCUS, TOPIC, TOPIC-REL), traverses the F-structure following the uncertainty paths. It computes probabilities for possible LDD resolutions, ranked by the product of the probabilities of the semantic forms and LDD paths. The highest ranked solution is returned.

7.2 Multilingual Treebank-Based LFG Grammar Acquisition

Cahill et al. (2002, 2004) and Cahill (2004) have presented a successful method for the treebank-based acquisition of rich, wide-coverage LFG resources for English. This raises the question whether it possible to apply this approach to other languages and treebank encodings.

The ParGram project (Butt et al., 2002) has succeeded in producing wide-coverage LFG grammars for a small number of languages (English, German, and Japanese, and smaller coverage grammars for French and Norwegian). Contrary to our approach, the ParGram grammars are hand-crafted, requiring a considerable amount of development time.

Cahill et al. (2003); Burke et al. (2004b); O'Donovan et al. (2005b), Cahill (2004) and Cahill et al. (2005) have provided early and preliminary proof-of-

concept research on the adaptation of the automatic F-structure annotation algorithm originally developed for English to Spanish, Chinese and German. Hockenmaier (2006) reports on the first steps on the automatic induction of rich CCG lexical resources for German. Hockenmaier transformed the TiGer treebank into a CCGbank and derived a wide-coverage CCG lexicon, but to date there are no parsing results for an automatically induced deep German CCG grammar.

The following section reviews previous work on LFG-based Grammar Acquisition for German, based on the early work by Cahill (2004) and Cahill et al. (2003, 2005).

7.3 Automatic Acquisition of Rich LFG Resources for German

Cahill (2004) and Cahill et al. (2003, 2005) develop an automatic F-structure annotation algorithm for the German TiGer treebank. They extract an F-structure-annotated PCFG Grammar from the F-structure-annotated TiGer treebank and present an evaluation of c-structure and F-structure parsing results against a manually constructed gold standard (DCU100) of 100 randomly extracted sentences from the TiGer treebank, and against 2000 automatically F-structure-annotated TiGer trees (CCG-style evaluation).

7.3.1 F-Structure Annotation and Evaluation for German

The automatic annotation of the TiGer treebank proceeds in a similar manner to the English annotation process. Out of the 40 000 sentences of the TiGer treebank, 96.9% receive one covering and connected F-structure, while 1112 sentences obtain more than one F-structure fragment. A small amount of sentences do not obtain any F-structure at all, due to feature clashes caused by inconsistencies in the annotation produced by the annotation algorithm.

Cahill (2004) evaluates the quality of the F-structures extracted from the original gold treebank trees against the DCU100, a manually created gold standard of 100 sentences randomly chosen from the TiGer treebank. These F-structures were converted into dependency structures adopting the method proposed by Forst (2003). The triple conversion and evaluation software of Crouch et al.

(2002) was used. Results (for gold treebank trees) show an overall F-score of 90.2% for preds-only, while the F-score for all grammatical functions is around 7% higher.

7.3.2 Parsing Experiments and Evaluation for German

For German Cahill (2004) and Cahill et al. (2003, 2005) performed parsing experiments, following the Integrated Model described in Section 7.1.3 above. Here I report results from Cahill (2004). The TiGer treebank was divided into a training set and a test set (sentences 8000-10000 of the TiGer treebank). The training set, which consists of all sentences of the TiGer treebank excluding the test set, was automatically annotated with F-structure equations. From the F-structure-annotated data an annotated PCFG (A-PCFG) was extracted, which then was used to parse the test set. A second version of the grammar was generated (PA-PCFG), using a parent transformation (Johnson, 1998) in addition to the F-structure annotations. The parser used in the experiments is BitPar (Schmid, 2004), an efficient parser for highly ambiguous context-free grammars. After parsing the test set with the A-PCFG and the PA-PCFG, the F-structure annotations present in the parse trees were collected and passed to a constraint solver, which generated F-structures from the equations.

Out of the 2000 sentences in the test set parsed with the A-PCFG, 95.5% received one covering and connected F-structure, while for the PA-PCFG for 97.9% of the sentences one covering and connected F-structure could be generated. The quality of the parsing results for raw text is evaluated in two ways: first against the manually created DCU100 gold standard and then against 2000 original TiGer treebank trees automatically annotated with F-structure equations (CCG-style evaluation). For constituent-based evaluation, Cahill (2004) reports an `evalb` labelled bracketing F-score of 69.4% on the parse trees generated with the A-PCFG against the original 2000 TiGer treebank trees, while the result for the PA-PCFG is slightly worse with 68.1%.

Evaluating the F-structures against the hand-crafted gold standard, Cahill (2004) achieves a labelled dependency F-score of 71% for the F-structures generated by the A-PCFG and 74.6% against the 2000 automatically annotated F-structures (CCG-style evaluation). For the PA-PCFG the results for the F-

structures are slightly worse than for the A-PCFG, with a decrease of 0.5% for the manually created gold standard and a decrease of 0.6% for the 2000 trees in the CCG-style evaluation. These results are in contrast to the effects of parent transformation for English, where parsing results improve (Johnson, 1998).

7.3.3 Parsing with Morphological Information

Morphological information plays an important role in German. While in English case assignment often uses configurational information, German makes use of its rich morphological system in order to determine specific grammatical functions such as subject, accusative object and so on. Therefore morphology could be a valuable source of information for the annotation process and for the disambiguation of parse trees, e.g. distinguishing the subject, which has to be in the nominative case, from the object in the accusative. Unfortunately the TiGer treebank (Version 1) does not include morphological annotation. In order to test the influence of morphological information on parsing results, Cahill (2004) simulates morphological information in the TiGer trees, using the functional labels in the TiGer trees. The subject (TiGer label SB) in German has to be in the nominative case, and the TiGer label OA indicates an accusative object. Automatically percolating this information down the head-projection in the TiGer tree and assigning it to the head nodes of the projection results in a TiGer treebank partly annotated with case information.

Two grammar transformations were used for the parsing experiments: an annotated grammar with case information (CA-PCFG) and a parent-transformed annotated PCFG with case information (CPA-PCFG), but none of them was able to improve the parsing results over the baseline reported in Section 2.5.2. As a possible reason for this somewhat unexpected result Cahill (2004) states that the simulation of case assignment was not fine-grained and accurate enough and therefore failed to support the parsing process.

7.4 Conclusions

Cahill et al. (2003), Cahill (2004) and Cahill et al. (2005) provide proof-of-concept, showing that the automatic acquisition of deep, wide-coverage probabilistic LFG

resources for German is possible in principle. After only three person months of development time they presented an automatically induced LFG grammar for German which achieved more than 95.7% coverage on unseen TiGer treebank data, while comparable hand-crafted grammars hardly exceed 70% (Forst, 2003), even after several years of development time. However, the work of Cahill et al. is limited in many ways. For evaluation purposes, Cahill (2004) and Cahill et al. (2003, 2005) could only revert to a hand-crafted gold standard of 100 sentences, which is too small to cover many of the interesting grammar phenomena present in the full TiGer data. The set of grammatical functions used for F-structure annotation was also rather small and coarse-grained, containing only 26 different features. Cahill et al. did not provide long-distance dependency resolution for parsing. In the remaining part of my thesis I present a substantially improved acquisition of deep, wide-coverage LFG resources for German.

Chapter 8

Improved Acquisition of Deep, Wide-Coverage LFG Resources for German: Preliminaries

8.1 Introduction

The remaining part of my thesis presents a significantly extended and improved method for the acquisition of deep, wide-coverage LFG resources for evaluating German, based on the early proof-of-concept work by Cahill et al. (2003); Cahill (2004); Cahill et al. (2005). This chapter describes the gold standard resources for evaluating treebank-based deep, wide-coverage LFG resources for German. I give an overview of different gold standards available for German, as well as the DCU250, a new gold standard I created for evaluating TiGer treebank-style F-structures.

8.2 Gold Standards for Evaluation

For German four dependency gold standards are now available for evaluation purposes: (1) the DCU100 (Cahill et al., 2003; Cahill, 2004), (2) the TiGer Dependency Bank (Forst, 2003; Forst et al., 2004) as well as an improved version of the TiGer DB, converted to XML (Boyd et al., 2007), (3) the DCU250 (my work) and, last but not least, (4) a small gold standard with 100 sentences from

DCU100		
governable functions	**non-govern. functions**	**atomic features**
adj-gen	adjunct	circ-form
adj-rel	app	comp-form
comp	app-clause	coord-form
obj	conj	part-form
obj2	dem	pron-type
obl	det	
obl-ag	name-mod	
obl-compar	number	
subj	poss	
xcomp	quant	
xcomp-pred		

Table 8.1: Grammatical functions in the DCU100

the TüBa-D/Z (Versley and Zinsmeister, 2006).[17] I will call this gold standard the TUBA100.

8.2.1 Gold Standards Based on the TiGer Treebank

The DCU100 was manually constructed by Cahill and Forst (Cahill et al., 2003; Cahill, 2004). They randomly extracted 100 sentences from a subset of the TiGer treebank (sentences 8000-10000). These 100 sentences were then converted into dependency structures following the method of Forst (2003) and manually corrected by Martin Forst. The DCU100 is restricted in two ways: its small size and also its small number of grammatical function and feature types. The DCU100 distinguishes only 26 different grammatical functions (Table 8.1), which is not sufficient to support a fine-grained analysis of linguistic phenomena.

The TiGer Dependency Bank (TiGer DB) (Forst, 2003; **?**) is much larger and provides a far more detailed, fine-grained annotation. It contains more

[17] Thanks to Yannick Versley and Heike Zinsmeister for providing the TüBa-D/Z gold standard.

TiGer DB		
governable functions	**non-govern. functions**	**atomic features**
cc	ams	case
da	app	circ-form
gl	app-cl	comp-form
gr	cj	coord-form
oa	cmpd-lemma	degree
obj	det	det-type
og	measured	fut
op	mo	gend
op-dir	mod	mood
op-loc	name-mod	num
op-manner	number	pass-asp
oc-inf	numverb	passive
oc-fin	pred-rest	perf
pd	quant	pers
sb	rc	precoord-form
sbp	rs	pron-form
	topic-disloc	pron-type
	topic-rel	tense

Table 8.2: Grammatical functions and features in the TiGer DB

DCU250		
governable functions	**non-govern. functions**	**atomic features**
adj-gen	adjunct	adjunct-type
adj-rel	ams	case
comp	app	circ-form
da	app-clause	comp-form
oa	conj	coord-form
oa2	det	degree
obj	measured	det-type
obj-gen	mod	fut
obl-compar	name-mod	gend
op	number	mood
pd	poss	num
sb	quant	part-form
sbp	rs	pass-asp
xcomp		perf
		pers
		postcoord-form
		precoord-form
		pron-form
		pron-type

Table 8.3: Grammatical functions and features in the DCU250

TUBA100		
governable functions	**non-govern. functions**	**atomic features**
cc	ams	case
da	app	comp-form
gl	app-cl	coord-form
gr	cfy	degree
oa	cj	det-type
obj	det	gend
op	fragment	mood
oc-inf	mo	num
oc-fin	name-mod	pass-asp
pd	rc	perf
sb		pron-type
		tense

Table 8.4: Grammatical functions and features in the TUBA100

than 1800 sentences of the TiGer treebank, semi-automatically converted into a dependency-based triple format using a large, hand-crafted LFG grammar for German (Dipper, 2003). With a set of 52 distinct grammatical functions and features (Table 8.2) it allows an in-depth description of different grammatical phenomena in German. However, there is one downside to the TiGer DB: it does not directly represent the actual surface tokens in the TiGer treebank. Resulting from the type of linguistic analysis adopted in the TiGer DB (which is based on the hand-crafted LFG grammar of Dipper (2003), it retokenises the TiGer strings, as for example for coordinations, for merged prepositions and determiners or for complex lexical items like compounds or pronominal adverbs. In other cases surface tokens have not been included in the analysis, as for *von*-PPs which function as phrasal genitives, where the preposition itself is not represented in the gold standard. Another case is the particle *zu* before infinitival verbs, which is dropped in the analysis. Substantial differences in tokenisation and linguistic analysis following the hand-crafted LFG grammar of Dipper (2003) make TiGer DB a problematic gold standard for the evaluation of TiGer treebank-trained and

machine-learning-based resources. The problems for evaluation are compounded by the fact that lemmatisation in the TiGer DB is largely based on the grammar of Dipper (2003) and does not follow the decisions made in the TiGer treebank. This means that in automatic evaluation of TiGer-treebank-based resources, in many cases a dependency representation is considered wrong, even if the correct analysis has been found.

This problem has been addressed by Boyd et al. (2007), who converted the TiGer DB into a more surface-oriented representation which allows us to match the dependency triples against the original treebank while preserving the rich linguistic information in the TiGer DB. The converted gold standard is encoded in a format called Decca-XML, which provides a flexible multi-purpose data structure, which can easily be adapted to different purposes.

However, there is a further major drawback with regard to the TiGer DB. Though it was created by transforming annotated trees from the TiGer treebank into dependency triples, in many cases the input from the TiGer treebank source does not provide enough information for the detailed description employed in the TiGer DB. The missing information was obtained by matching the converted TiGer DB trees against the output of a hand-crafted, broad-coverage LFG grammar (Dipper, 2003). This leads to a many-to-many mapping between the functional labels in the TiGer treebank and the corresponding grammatical features annotated in the TiGer DB representing the richer annotations in the hand-crafted grammar of Dipper (2003): for example, modifiers (MO) in the TiGer treebank can either obtain the annotation modifier (mo), predicate (pd), oblique directional argument (op_dir), or oblique local argument (op_loc) in the TiGer Dependency Bank. Modifiers (mo) in the TiGer DB, on the other hand, can be encoded as modifiers (MO), appositions (APP), as a measure argument of an adjective (AMS) or a comparative complement (CC) in the TiGer treebank. For evaluating machine-learning- and treebank-based grammar acquisition methods, this makes a mapping between TiGer DB and TiGer- and machine-learning-based resources very difficult, and in fact strongly biases TiGer DB-based evaluation in favour of the hand-crafted LFG grammar of Dipper (2003). In order to support a fair evaluation, I created another gold standard of 250 sentences from the TiGer treebank, randomly chosen from sentences 8000-10000. The DCU250 uses a set of 45 different grammatical functions and features (Table 8.3), encoding

only information which can actually be induced from the TiGer treebank.

The Creation of the DCU250

The feature set of the DCU250 (Table 8.3) is not as detailed as the one in the TiGer DB (Table 8.2), but it is substantially more fine-grained than the one in the DCU100 (Table 8.1), and it only encodes information which can be directly or implicitly derived from the TiGer treebank.

The creation of the DCU250 for 250 sentences randomly selected from the TiGer treebank used the original F-structure annotation algorithm of Cahill et al. (2003) and Cahill (2004) for German: I roughly adapted the F-structure annotation algorithm to the new feature set, while accepting a certain amount of noise and errors. I used the algorithm to automatically generate dependency triples for the sentences of the DCU250. Then I manually corrected and extended these triples to produce the DCU250.

8.2.2 A Gold Standard Based on the TüBa-D/Z

The TUBA100 was semi-automatically created by Heike Zinsmeister and Yannick Versley, using the conversion method of Versley (2005) on 100 randomly selected gold trees from the TüBa-D/Z. Versley's conversion method uses a set of hand-crafted rules that transform the original TüBa-D/Z annotations to dependencies, following the format of the Weighted Constraint-Based Dependency Parser (WCDG) (Foth et al., 2004). The converted output was then adapted to a set of grammatical features (Table 8.4) maximally similar to the TiGer DB. This is a great advantage for evaluation, because it allows us to compare not only different LFG grammar acquisition architectures, but also results for different treebank annotation schemes.

In Section 8.2.1 I discussed the problems caused by the restricted size of the DCU100. These problems also apply here. Even though the TUBA100 was adapted to the fine-grained set of grammatical features used in the TiGer DB, due to its size the TUBA100 cannot cover all relevant grammatical phenomena in German and, as it was used for development of the F-structure annotation algorithm on the TüBa-D/Z, the evaluation results of the automatic annotation are expected to be less reliable and the overall annotation coverage on TüBa-D/Z

trees will be lower than the one for TiGer trees.

8.3 Summary

This chapter described four different gold standards based on the TiGer and TüBa-D/Z treebanks and discussed their adequacy for the evaluation of automatically acquired LFG resources.

In the next Chapter I develop different versions of an F-structure annotation algorithm for German for TiGer and TüBa-D/Z, and adapted to three of the gold standards described in Chapter 8, namely the TiGer DB, DCU250 and TUBA100.

Chapter 9

Developing F-structure Annotation Algorithms for German

9.1 Introduction

This chapter describes the development of F-structure annotation algorithms for German, based on the feature sets in the TiGer DB, DCU250 and TUBA100 gold standards. I highlight the differences to the English LFG grammar acquisition architecture described in Chapter 7, caused by the language-specific properties of German, which are reflected in the differences between tree structures in the English Penn-II treebank and the German TiGer and TüBa-D/Z treebanks. Finally, I present results for automatic F-structure annotation on gold trees for TiGer and TüBa-D/Z and the TiGerDB, DCU250 and TUBA100 gold standards.

9.2 Developing F-Structure Annotation Algorithms for the Extended Feature Sets in the TiGer DB, DCU250 and TUBA100

Before developing annotation algorithms for each of the three gold standards I divided the TiGer DB into a development set of 1366 sentences and a test set of

500 sentences. I did the same for the DCU250, but due to the smaller size of the newly created gold standard, the development set and the test set consist of 125 sentences each. The TUBA100 is too small to be split, so I used all 100 sentences for both development and testing. It is understood that a larger data set would be more appropriate, and that the use of the same data for development and testing may skew results. Section 9.3 reports results both on the development sets and on the test sets for the TiGer-based gold standards. For the TüBa-D/Z I give results on the development set only.

The development of the F-structure annotation algorithm for the extended set of features in the TiGer DB is by no means a straightforward process. Besides the many-to-many mapping between grammatical functions in both the TiGer and TiGer DB encoding schemes, the treatment of auxiliary verbs is another major source of problems. Following the hand-crafted German LFG grammar of Dipper (2003), in the TiGer DB auxiliaries are not assumed to have a subcategorisation frame but are rather treated as feature-carrying elements, expressing information about tense or aspect. This reflects their different status in comparison to modals or other raising verbs (Butt et al., 1996). While this annotation style is based on a thorough linguistic analysis and avoids unnecessary structural complexity, it is not consistent with the annotation in the TiGer treebank, where auxiliaries are annotated as the head of the sentence. This means that for an evaluation against the TiGer DB the TiGer treebank-style annotation of auxiliaries has to be converted to TiGer DB-style, removing the predicates of the auxiliaries from the F-structure while preserving the grammatical features expressed by the auxiliaries. However, there are many cases where the extraction of these features cannot be disambiguated easily.

One example concerns cases where the auxiliary *sein* (to be) is combined with a past participle. This construction can either be a Stative Passive, a predicative argument or a form of the German Perfekt.[18] The annotation in the TiGer treebank (and also the one in the TüBa-D/Z) does not provide enough information to distinguish between these constructions.

Only for impersonal passive constructions does the TiGer treebank annotation reveal the deep grammatical functions of the constituents. In all other

[18]See also Maienborn (2007) for an analysis of *sein* + past participle as a copula along with the adjectivisation of the past participle.

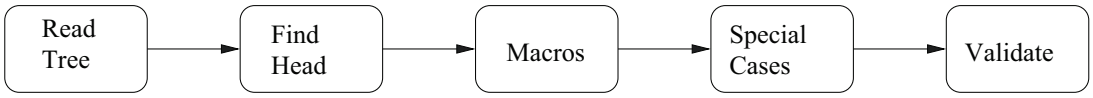


Figure 9.1: The modules of the AA

cases the linguistic function of the construction in question has to be decided on-the-fly, while the information required for disambiguation is not provided in the TiGer treebank. In order to solve these ambiguities, valency information is needed: intransitive verbs do not allow for passivisation. Therefore I automatically extracted subcategorisation frames for all verbs in the TiGer treebank, which helped to improve the correct annotation of the grammatical features for the Stative Passive, the German Perfekt and for predicative arguments. However, even for a treebank with 50 000 sentences the results still suffer from data sparseness and can be improved by a larger coverage valency dictionary. The Constraint Dependency Grammar (CDG) (Foth et al., 2004) provides such a dictionary with entries for more than 8200 verbs. I include the CDG valency dictionary in the annotation algorithm as an external source of knowledge, helping to disambiguate between Stative Passive and German Perfekt constructions.

The LFG F-structure annotation algorithm (AA) for English and the early preliminary work for the German TiGer treebank (Cahill et al., 2003; Cahill, 2004; Cahill et al., 2005) was implemented in Java. I reimplemented the AA in Perl, which combines object-oriented features with powerful handling of regular expressions. In contrast to the original AA, which was working on Penn-II-style treebank trees, my implementation of the annotation algorithm takes trees in the NEGRA export format (Skut et al., 1997) as input.

My German LFG AA proceeds as follows (Figure 9.1): first it reads in the treebank trees encoded in the NEGRA export format and converts each tree into a tree object. Then it applies head-finding rules (Table 9.1) which I developed for TiGer in the style of Magerman (1995), in order to determine the head of each local node.[19] The head-finding rules specify a set of candidate heads, depending

[19]TiGer provides head annotation for all categorial nodes except NPs, PPs and PNs. Due to the flat annotation in TiGer, partly resulting from the decision not to annotate unary nodes, the problem of identifying the correct head for those nodes is more severe than for the TüBa-D/Z, where the more hierarchical structure results in smaller constituents which, in addition, are all

on the syntactic category of the node, and also the direction (left/right) in which the search should proceed. For prepositional phrases, for example, we start from the left and look at all child nodes of the PP. If the left-most child node of the PP has the label KOKOM (comparative particle), we assign it the head of the PP. If not, we check if it is a preposition (APPR), a preposition merged with a determiner (APPRART), an apposition (APPO), and so on. If the left-most child node does not carry one of the candidate labels listed in Table 9.1, we take a look at the next child node, working our way from left to right.

For some of the nodes these head-finding rules work quite well, while for others we have to accept a certain amount of noise. This is especially true for the flat NPs in the TiGer treebank. A *Special Cases* module checks these nodes at a later stage in the annotation process and corrects possible errors made in the annotation.

After determining the heads, the tree is handed over to the *Macros* module which assigns F-structure equations to each node. This is done with the help of macros. Sometimes these macros overgeneralise and assign an incorrect grammatical function. In order to deal with this, the *Special Cases* module corrects inappropriate annotations made by the *Macros* module. Finally the *Validation* module takes a final look at the annotated trees and makes sure that every node has been assigned a head and that there is no node with two child nodes carrying the same governable grammatical function.

9.2.1 Differences between the English and the German Annotation Algorithm

The most important difference in the design of the English and the German AAs concerns the application of left-right context rules in the English annotation algorithm. These rules express annotation generalisations and have been handcrafted by looking at the most frequent grammar rules for each node in the Penn-II treebank and are also applied to unseen low-frequency rules. A sample partial

head-marked. When annotation original treebank trees, the head-finding rules are applied to NP, PP and PN nodes, when running the AA on parser output trees with erroneous or no GF labels in the trees, I also make use of head-finding rules for other syntactic categories (see Table 9.1)

Category	Direction	Values
AA	right	ADJD PIS PIAT ADV ADJA
AP	right	ADJA ADJD CARD ART PIAT NN PIS ADV PDAT VVPP PTKNEG PWAT TRUNC
AVP	right	ADV PTKNEG PROAV PWAV ADJD PWAT PIS PTKA PIAT APPR KOUS PTKANT KON KOUS NN
CAC	right	KON
CAP	right	KON APPR ADV
CAVP	right	KON APPR
CCP	right	KON
CH	right	NN NE FM CARD XY KON ADV ITJ
CNP	right	KON
CO	right	KON APPR ADV KOKOM PROAV
CPP	right	KON ADV
CS	right	KON ADV
CVP	right	KON
CVZ	right	KON
DL	right	NE NN KON ADV NP PP PN CNP S CS
ISU	left	ADV APPR KON PIS
MTA	right	ADJA NE NN
NM	right	NN CARD ADJA
NP	left	NN NE PPER FM PIS PDS PWS PRELS PRF PPOSS CH CNP NP PIAT PN CARD AP ADJA ART
PN	right	NE NNE NN NP CNP
PP	left	KOKOM APPR APPRART APPO PROAV APZR KOUS NE FM PDS
QL	right	CARD
S	left	VAFIN VMFIN VVFIN VVIMP VAIMP VVPP VAINF VMINF VVFIN VVIZU
VP	left	VVPP VVINF VAINF VMINF VAPP VMPP VVIZU VVFIN VMFIN VZ CVZ CVP ADJD TRUNC PP
VZ	right	VVINF VMINF VAINF ADJA VVIZU

Table 9.1: Head-finding rules for the TiGer treebank

left-context	head	right-context
JJ, ADJP: $\downarrow = \in \uparrow$ ADJUNCT	NN, NNS, ... $\uparrow=\downarrow$	NP: $\downarrow = \in \uparrow$ APP

Table 9.2: Left-right context rule used in the English AA

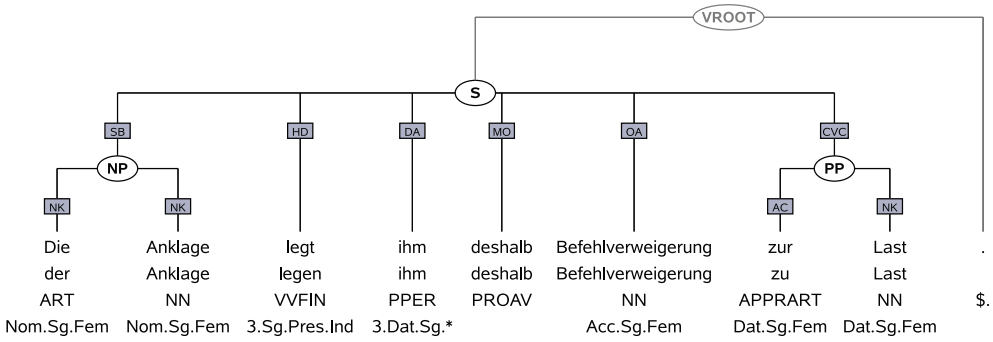


Figure 9.2: TiGer treebank tree example for free word order in German

left-right-context rule for NPs is given in Table 9.2.

The left-context rule states that all adjectives or adjectival phrases to the left of the head of an NP should be annotated as an adjunct, while the right-context rule specifies that an NP to the right of the head of an NP is an apposition. The creation of these left-right-context rules needs linguistic expertise and crucially depends on configurational properties of English.

For English, these rules successfully specify the correct annotation for the majority of local nodes in a given tree. For German, however, these rules do not work as well as for English. Table 9.3 illustrates this point by showing different possibilities for the surface realisation of a (rather short) German sentence (24).

(24) Die Anklage legt ihm deshalb Befehlsverweigerung zur Last.
the prosecution lies him therefore refusal to obey to the burden.
The prosecution therefore charges him with the refusal to obey.

Table 9.3 shows the variability of word order in German. The F-structure-annotated grammar rule for S in Figure 9.3 tells us that the first NP *Die An-*

S	→	NP	VVFIN	PPER	PROAV	NN	PP
		↑ SUBJ=↓	↑=↓	↑ DA=↓	↓∈↑ MO	↑ OA=↓	↑ OP=↓

Figure 9.3: F-structure equations for the grammar rule in Figure 9.2

klage (the prosecution) is the subject of the sentence, while the noun *Befehlsverweigerung* (refusal to obey) should be annotated as an accusative object, and the pronominal adverb *deshalb* (therefore) is an element of the modifier set. Table 9.3, however, illustrates that these constituents can occur in very different positions to the left or right of the head of the sentence. This shows that, unlike for a strongly configurational language such as English, the specification of left-right-context rules for German is not very helpful.

Instead of developing horizontal and strongly configurational context rules, my AA for German makes extended use of macros, using different combinations of information such as part-of-speech tags, node labels, edge labels and parent node labels (as encoded in the TiGer and TüBa-D/Z treebanks). First I apply more general macros assigning functional annotations to each POS, syntactic category or edge label in the tree. More specific macros such as the combination of a POS tag with the syntactic node label of the parent node, or a categorial node with a specific grammatical function label, can overwrite these general macros. The order of these macros is crucial, dealing with more and more specific information. Some of the macros overwrite information assigned before, while others only add more information to the functional annotation.

To give an example, consider the POS tag ART (determiner). The first macro is triggered by this POS tag and assigns the F-structure equation $\uparrow{=}\downarrow, \downarrow$ *det-type* $= def$. The next macro looks at combinations of POS tags and grammatical function (GF) labels and, for a determiner with the label NK (noun kernel), adds the equation $\uparrow$ *spec* : *det* $=\downarrow$, while the same POS tag gets assigned the functional equation $\downarrow{\in}\uparrow$ *spec* : *number* when occurring with the edge label NMC (numerical component). The annotation for the combination of POS and grammatical function label can be overwritten when a more specific macro applies, e.g. one which also considers the parent node for a particular POS-GF-combination.

The determiner with edge label NK has so far been annotated with *headword*, $\downarrow$ *det-type* $= def$, $\uparrow$ *spec* : *det* $=\downarrow$. This is overwritten with the F-structure equation

Die Anklage	legt	ihm	deshalb	Befehlsverweigerung	zur Last.
Die Anklage	legt	deshalb	Befehlsverweigerung	ihm	zur Last.
Die Anklage	legt	deshalb	ihm	Befehlsverweigerung	zur Last.
Die Anklage	legt	deshalb	ihm	zur Last	Befehlsverweigerung.
Befehlsverweigerung	legt	ihm	deshalb	die Anklage	zur Last.
Befehlsverweigerung	legt	deshalb	ihm	die Anklabe	zur Last.
Befehlsverweigerung	zur Last	legt	ihm	deshalb	die Anklage.
Befehlsverweigerung	zur Last	legt	deshalb	ihm	die Anklage.
Befehlsverweigerung	zur Last	legt	deshalb	ihm	die Anklage.
Ihm	legt	die Anklage	deshalb	Befehlsverweigerung	zur Last.
Ihm	zur Last	legt	deshalb	die Anklage	Befehlsverweigerung.
Ihm	zur Last	legt	die Anklage	deshalb	Befehlsverweigerung.
Zur Last	legt	ihm	deshalb	die Anklage	Befehlsverweigerung.
Zur Last	legt	ihm	die Anklage	deshalb	Befehlsverweigerung.
Zur Last	legt	die Anklage	ihm	deshalb	Befehlsverweigerung.
Deshalb	legt	ihm	die Anklage	Befehlsverweigerung	zur Last.
...	...	...	...	...	...

Table 9.3: Example for variable word order in German

$\uparrow obj : spec : det = \downarrow$, if it is the child of a PP node. This is due to the fact that the annotation guidelines of the TiGer treebank analyse prepositions as the head of a PP, while the head noun (and its dependents) inside the PP is annotated as the object of the preposition.

Due to the flat annotation in TiGer, it is not helpful to use vertical context above parent node level. The AA makes heavy use of the *Special Cases* module, where further annotation rules are specified for most syntactic categories. One tricky case is that of NPs, which have a totally flat structure in the TiGer treebank. There are many cases where the information about POS tag and grammatical function label is not sufficient, and neither is their relative position to the head of the phrase. In those cases the presence or absence of other nodes decides the grammatical function of the node in question.

To illustrate this, consider the three examples in Figures 9.4-9.6. All three examples show an NP with a noun child node followed by a proper name (PN) node, but where the grammatical annotations differ crucially. In Figure 9.4, the PN is the head of the NP. In Figure 9.5, where we have a determiner to the left of the noun (NN), the noun itself is the head of the NP, while the PN is an apposition. The third example (Figure 9.6) looks pretty much like the second one, with the exception that *Merkel* is in the genitive case. Here the PN should be annotated as a genitive attribute. This is not so much a problem for the annotation of the original treebank trees where we have both the correct grammatical function labels as well as morphological information. For parser output, however, morphological information is not available and the grammatical functions assigned are often incorrect.

Compared to the TiGer DB, the reimplementation of the F-structure Annotation Algorithm for the DCU250 was less problematic, because the grammatical features used in the DCU250 are designed to match the functional labels in the TiGer treebank. However, problems like the ones described above also apply here.

9.2.2 Differences between the New AA for German and Cahill et al. (2003, 2005) and Cahill (2004)

The annotation algorithm for German presented in this chapter is based on and substantially revises and extends preliminary work by Cahill et al. (2003, 2005)

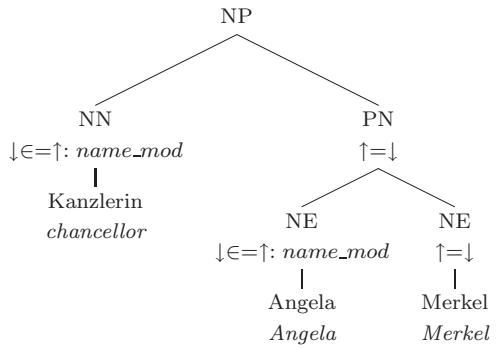


Figure 9.4: NP-internal structure in TiGer (PN=head)

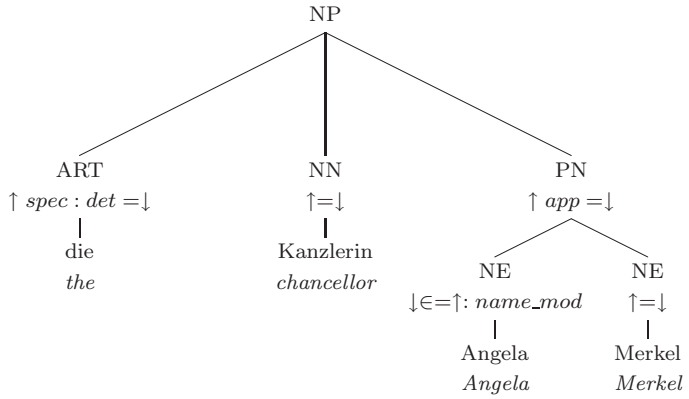


Figure 9.5: NP-internal structure in TiGer (PN=apposition)

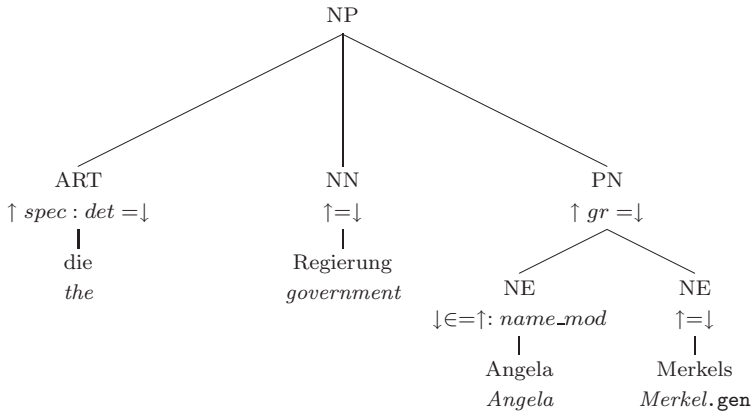


Figure 9.6: NP-internal structure in TiGer (PN=genitive to the right)

and Cahill (2004). The AA by Cahill et al. provides annotations for a rather limited set of grammatical functions only (see Chapter 8). The annotation of the German TiGer treebank as presented by Cahill et al. is a two-stage process, where in the first stage the AA tries to assign a default LFG equation to each node in the tree, based on the TiGer grammatical function label assigned to the node. As this often overgenerates and results in incorrect annotations, in a second stage the AA overwrites the default annotations for certain grammatical constructions. These include the identification of PP objects, the behaviour of complementisers, as well as determining the head of a coordination phrase with more than one coordinating conjunction. Finally, in a post-processing stage, the AA links trace nodes present in the Penn treebank-style version of the TiGer treebank to their reference nodes.

In my work I use a substantially extended set of grammatical functions and features, as described in Chapter 8. As a result, the annotated resources contain richer linguistic information and are of higher quality and usefulness compared to the one of Cahill et al. (2003, 2005) and Cahill (2004). I extend the default annotations triggered by the grammatical function labels in TiGer and define a set of macros using vertical context information in the trees, like the syntactic category or the grammatical function label of the node and its parent node, and combinations of both (see Section 9.2.1). My annotation algorithm also makes use of a valency dictionary in order to distinguish between stative passive constructions and the German Perfekt with *sein* (to be). In contrast to Cahill et al. (2003, 2005) and Cahill (2004), who work on Penn-II-style TiGer (Release 1) treebank trees, a converted, context-free version of the original TiGer graph structure, my version of the annotation algorithm takes trees in the NEGRA export format (Skut et al., 1997) as input. Therefore the post-processing stage for linking trace nodes with their corresponding reference nodes becomes unnecessary.

The next section reports on evaluation results for automatic F-structure annotation of gold treebank trees.

9.3 Results for Automatic F-structure Annotation on Gold Trees

This section reports evaluation results for the automatic F-structure annotation on original TiGer/ TüBa-D/Z treebank trees for

1. TiGer DB-style F-structures evaluated against the TiGer DB;
2. TiGer treebank-style F-structures evaluated against the DCU250;
3. and TüBa-D/Z-style F-structures (similar to the TiGer DB) evaluated against the TUBA100.

In the experiments I use a slightly modified version of the TiGer DB, with the following changes:

- The fine-grained annotation of *op-loc, op-dir, op-manner* cannot be induced automatically from the TiGer treebank. Therefore I merged all three functions into the grammatical function *op*.
- The TiGer DB decomposes German compound words (i.e. it retokenises the TiGer treebank data). The AA does not include a morphological analyser, therefore I recomposed the compounds and treat them like regular nouns. Due to the lack of a morphological analyser, I only include morphological features in the evaluation of the AA on gold treebank trees. For the annotation and evaluation of raw text (i.e. parser output in Chapter 10) these features are excluded.

For TiGer DB recall (all grammatical functions) is 84.8%, while precision is notably higher with 87.8% (Table 9.4). 99.8% of the trees produce one covering and connected F-structure; 3 out of the 1866 gold trees did not receive an F-structure, due to clashes caused by inconsistencies in the annotation. The results reflect the problems described above, caused by the many-to-many mapping of grammatical functions between the TiGer treebank and the TiGer DB and the lack of information in the TiGer treebank needed for the fine-grained annotation in the TiGer DB. Results for the DCU250 test set, in comparison, are significantly higher with a precision of 96.8% and a recall of 97.5%. Only one out of the 250 sentences did not receive an F-structure.

Not surprisingly, results for the development sets for both annotation styles are slightly higher with 97.8% (precision) and 98.1% (recall) for the DCU250 development set and the same precision, but a higher recall of 86.7% for the TiGer DB development set. Results for the TUBA100 are lower than for the DCU250 (precision: 95.5%, recall: 94.6%), but significantly higher than for the TiGerDB. Two sentences in the TUBA100 did not receive an F-structure.

Detailed results broken down by grammatical functions are provided in Tables 9.5, 9.6, 9.7, 9.8 and 9.9. Results for the DCU250 (Tables 9.5 and 9.6) are quite high for most dependency relations and features. Incorrect assignments mostly arise where the dependency relation or grammatical feature cannot be induced from the GF label in the treebank (e.g. numbers (number), name modifiers (name-mod) or quantifiers (quant)). For the TiGer DB (Tables 9.7 and 9.8) we also observe low results for cases where the grammatical function label in the TiGer treebank can be mapped to more than one dependency relation in the TiGer DB, and vice versa (e.g. appositions (app), modifiers (mo), predicates (pd)). Another difficult case is low-frequency dependency relations (e.g. reported speech (rs)). As a result, F-scores for the TiGer DB data sets are significantly lower than for the DCU250.

Results for the TüBa-D/Z (Table 9.9) reflect a problem specific to the annotation of non-local dependencies in the treebank: head and dependent often end up in different topological fields, and it is non-trivial to recover the correct dependencies, especially if they are labelled as MOD (ambiguous modifier). In those cases the correct dependency can only be guessed. Another problem caused by the design of the TüBa-D/Z is the annotation of appositions (app) (see Section 5.3.1, Figure 5.5), which also leads to low results in the F-structure evaluation. The results presented here using "perfect" treebank trees with full morphological and functional information provide upper bounds for the parsing experiments reported in the next chapter.

9.4 Summary

In this chapter I described the development of different versions of an F-structure annotation algorithm for German, based on different treebanks and gold standard resources. I discussed problems arising through language-specific properties

	development set			*test set*		
AA-style	**Prec.**	**Rec.**	**F-score**	**Prec.**	**Rec.**	**F-score**
TiGerDB	87.8	86.7	87.3	87.8	84.8	86.3
DCU250	97.8	98.1	97.9	96.8	97.5	97.1
TUBA100	95.5	94.6	95.0			

Table 9.4: Results for automatic F-structure annotation on gold trees

of German like the semi-free word order, which is reflected in the flat tree structure annotated in the TiGer treebank and the topological fields in TüBa-D/Z, and showed how the problem can be addressed by applying macros encoding different combinations of local information from syntactic node labels, grammatical function labels and POS tags.

Evaluating automatic F-structure annotations on gold treebank trees from the TiGer and TüBa-D/Z treebanks shows that the different versions of the annotation algorithm yield satisfactory results on the DCU250 and the TUBA100 test sets. Lower results for the TiGer DB test set, compared to the DCU250, are due to (i) the more fine-grained linguistic information annotated in the gold standard which cannot be automatically induced from the TiGer treebank, and (ii) to many-to-many mapping problems between TiGer and the TiGer DB.

The next chapter reports on parsing experiments with PCFGs extracted from the TiGer and TüBa-D/Z treebanks, annotated in the TiGer DB style, the DCU250 style and the TUBA100 style, respectively.

DEPENDENCY		Precision		Recall	F-Score
adj-gen	100	(104/104)	100	(104/104)	100
adj-rel	100	(25/25)	100	(25/25)	100
ams	100	(1/1)	100	(1/1)	100
app	95	(55/58)	95	(55/58)	95
app-clause	100	(10/10)	100	(10/10)	100
circ-form	100	(2/2)	100	(2/2)	100
comp	96	(22/23)	96	(22/23)	96
comp-form	92	(12/13)	86	(12/14)	89
conj	96	(190/197)	95	(190/201)	95
coord-form	100	(73/73)	99	(73/74)	99
da	100	(8/8)	100	(8/8)	100
degree	98	(259/263)	99	(259/261)	99
det	100	(421/423)	99	(421/426)	99
det-type	100	(421/421)	100	(421/421)	100
fut	100	(11/11)	100	(11/11)	100
gend	100	(834/838)	100	(834/836)	100
measured	100	(3/3)	100	(3/3)	100
mo	95	(675/712)	95	(675/713)	95
mo-type	100	(22/22)	100	(22/22)	100
mod	50	(1/2)	50	(1/2)	50
mood	97	(214/221)	100	(214/214)	98
name-mod	89	(41/46)	98	(41/42)	93
num	98	(1115/1134)	100	(1115/1120)	99
number	77	(24/31)	86	(24/28)	81
oa	98	(97/99)	98	(97/99)	98
obj	98	(342/350)	98	(342/349)	98
obj-gen	100	(1/1)	100	(1/1)	100
obl-compar	100	(10/10)	100	(10/10)	100
op	97	(36/37)	97	(36/37)	97
part-form	100	(14/14)	100	(14/14)	100
pass-asp	100	(29/29)	100	(29/29)	100
pd	100	(37/37)	92	(37/40)	96
perf	100	(27/27)	100	(27/27)	100
pers	96	(262/272)	99	(262/265)	98
poss	100	(26/26)	96	(26/27)	98
postcoord-form	100	(1/1)	100	(1/1)	100
pron-form	100	(8/8)	100	(8/8)	100
pron-type	96	(117/122)	96	(117/122)	96
quant	98	(44/45)	98	(44/45)	98
rs	0	(0/0)	0	(0/2)	0
sb	95	(299/316)	93	(299/320)	94
sbp	100	(6/6)	100	(6/6)	100
tense	97	(214/221)	100	(214/214)	98
tiger-id	100	(131/131)	98	(131/134)	99
xcomp	95	(40/42)	100	(40/40)	98
RESULT:	**97.8**		**98.1**		**97.9**

Table 9.5: Results for automatic F-structure annotation on gold trees (DCU250 development set)

DEPENDENCY		Precision		Recall	F-Score
adj-gen	100	(70/70)	99	(70/71)	99
adj-rel	93	(14/15)	93	(14/15)	93
ams		(0/0)		(0/0)	
app	87	(27/31)	93	(27/29)	90
app-clause	100	(6/6)	100	(6/6)	100
case	99	(643/647)	100	(643/646)	99
circ-form	100	(3/3)	100	(3/3)	100
comp	100	(17/17)	100	(17/17)	100
comp-form	100	(9/9)	100	(9/9)	100
conj	97	(154/158)	97	(154/159)	97
coord-form	100	(63/63)	97	(63/65)	98
da	100	(11/11)	100	(11/11)	100
degree	99	(164/165)	99	(164/165)	99
det	99	(298/302)	98	(298/305)	98
det-type	99	(299/301)	98	(299/304)	99
fut	100	(5/5)	100	(5/5)	100
gend	99	(586/589)	99	(586/589)	99
measured	100	(1/1)	100	(1/1)	100
mo	93	(458/495)	94	(458/487)	93
mo-type	100	(13/13)	100	(13/13)	100
mod	100	(4/4)	100	(4/4)	100
mood	96	(188/195)	99	(188/189)	98
name-mod	66	(23/35)	92	(23/25)	77
num	98	(828/846)	99	(828/833)	99
number	74	(35/47)	85	(35/41)	80
oa	97	(85/88)	92	(85/92)	94
oa2	100	(1/1)	100	(1/1)	100
obj	98	(238/243)	98	(238/244)	98
obj-gen	100	(1/1)	100	(1/1)	100
obl-compar	100	(4/4)	100	(4/4)	100
op	100	(28/28)	90	(28/31)	95
part-form	95	(18/19)	100	(18/18)	97
pass-asp	97	(28/29)	93	(28/30)	95
pd	96	(24/25)	100	(24/24)	98
perf	88	(21/24)	95	(21/22)	91
pers	96	(244/255)	99	(244/246)	97
poss	100	(16/16)	100	(16/16)	100
pred-restr	100	(4/4)	100	(4/4)	100
pron-form	100	(2/2)	50	(2/4)	67
pron-type	93	(84/90)	89	(84/94)	91
quant	90	(18/20)	86	(18/21)	88
sb	91	(231/253)	94	(231/247)	92
sbp	100	(2/2)	100	(2/2)	100
tense	95	(186/195)	99	(186/188)	97
tiger-id	100	(139/139)	99	(139/140)	100
xcomp	100	(30/30)	100	(30/30)	100
RESULT:	**96.8**		**97.5**		**97.1**

Table 9.6: Results for automatic F-structure annotation on gold trees (DCU250 test set)

DEPENDENCY	Precision		Recall		F-Score
ams	64	(7/11)	78	(7/9)	70
app	52	(253/484)	83	(253/306)	64
app-cl	69	(57/83)	89	(57/64)	78
cc	45	(25/56)	61	(25/41)	52
circ-form	46	(6/13)	100	(6/6)	63
cj	92	(1447/1573)	91	(1447/1592)	91
comp-form	94	(111/118)	86	(111/129)	90
coord-form	98	(570/579)	96	(570/594)	97
da	92	(110/119)	92	(110/119)	92
det	96	(3369/3512)	95	(3369/3541)	96
det-type	96	(3343/3483)	98	(3343/3400)	97
fut	97	(56/58)	95	(56/59)	96
gl	96	(218/228)	90	(218/241)	93
gr	82	(681/831)	80	(681/853)	81
measured	88	(14/16)	88	(14/16)	88
mo	82	(4799/5849)	81	(4799/5917)	82
mod	94	(31/33)	94	(31/33)	94
name-mod	76	(346/458)	95	(346/364)	84
number	67	(217/325)	55	(217/398)	60
numverb	0	(0/0)	0	(0/6)	0
oa	93	(852/916)	91	(852/936)	92
oa2	0	(0/1)	0	(0/0)	0
obj	91	(2702/2981)	93	(2702/2919)	92
oc-fin	89	(157/176)	84	(157/188)	86
oc-inf	86	(353/412)	86	(353/410)	86
og	86	(6/7)	86	(6/7)	86
op	90	(509/563)	85	(509/597)	88
pass-asp	88	(256/290)	81	(256/318)	84
passive	0	(0/0)	0	(0/2)	0
pd	80	(211/263)	62	(211/341)	70
perf	99	(226/228)	79	(226/286)	88
precoord-form	100	(8/8)	100	(8/8)	100
pred-restr	0	(0/0)	0	(0/1)	0
pron-form	98	(50/51)	93	(50/54)	95
pron-type	75	(724/969)	68	(724/1061)	71
quant	91	(124/137)	66	(124/187)	77
rc	91	(175/193)	83	(175/210)	87
rs	0	(0/0)	0	(0/3)	0
sb	85	(2255/2652)	79	(2255/2862)	82
sbp	80	(45/56)	74	(45/61)	77
tiger-id	93	(1138/1224)	94	(1138/1207)	94
topic-disloc	0	(0/0)	0	(0/2)	0
topic-rel	0	(0/0)	0	(0/1)	0
total	**87.8**		**86.7**		**87.3**

Table 9.7: Results for automatic F-structure annotation on gold trees (TiGer DB development set)

DEPENDENCY		Precision		Recall	F-Score
ams	0	(0/5)	0	(0/2)	0
app	53	(65/122)	66	(65/98)	59
app-cl	58	(22/38)	85	(22/26)	69
cc	38	(11/29)	52	(11/21)	44
circ-form	25	(1/4)	100	(1/1)	40
cj	96	(597/625)	91	(597/654)	93
comp-form	87	(52/60)	73	(52/71)	79
coord-form	98	(212/217)	94	(212/225)	96
da	81	(57/70)	79	(57/72)	80
det	96	(1148/1195)	94	(1148/1223)	95
det-type	98	(1163/1185)	97	(1163/1194)	98
fut	100	(27/27)	93	(27/29)	96
gl	93	(98/105)	92	(98/106)	93
gr	83	(196/237)	79	(196/249)	81
measured	100	(7/7)	88	(7/8)	93
mo	77	(1715/2226)	80	(1715/2140)	79
mod	100	(11/11)	100	(11/11)	100
name-mod	77	(99/129)	97	(99/102)	86
number	70	(88/125)	63	(88/139)	67
numverb	0	(0/0)	0	(0/2)	0
oa	92	(381/415)	87	(381/440)	89
oa2	100	(1/1)	100	(1/1)	100
obj	91	(964/1058)	92	(964/1047)	92
oc-fin	79	(59/75)	75	(59/79)	77
oc-inf	88	(127/145)	93	(127/136)	90
og	100	(2/2)	100	(2/2)	100
op	86	(131/153)	48	(131/275)	61
pass-asp	92	(70/76)	80	(70/87)	86
pd	82	(96/117)	66	(96/146)	73
perf	99	(104/105)	84	(104/124)	91
precoord-form	100	(5/5)	100	(5/5)	100
pred-restr	0	(0/0)	0	(0/1)	0
pron-form	100	(27/27)	84	(27/32)	92
pron-type	83	(439/530)	72	(439/607)	77
quant	87	(58/67)	60	(58/96)	71
rc	95	(70/74)	80	(70/87)	87
rs	0	(0/0)	0	(0/1)	0
sb	88	(948/1076)	81	(948/1175)	84
sbp	100	(15/15)	100	(15/15)	100
tiger-id	96	(387/405)	93	(387/414)	95
topic-disloc	0	(0/0)	0	(0/2)	0
total	**87.8**		**84.8**		**86.3**

Table 9.8: Results for automatic F-structure annotation on gold trees (TiGer DB test set)

DEPENDENCY		Precision		Recall	F-Score
ams	100	(1/1)	100	(1/1)	100
app	64	(9/14)	50	(9/18)	56
app_cl	0	(0/0)	0	(0/1)	0
case	98	(497/506)	97	(497/510)	98
cc	100	(3/3)	60	(3/5)	75
cj	87	(112/129)	93	(112/120)	90
comp_form	83	(5/6)	100	(5/5)	91
coord_form	90	(35/39)	90	(35/39)	90
da	100	(4/4)	100	(4/4)	100
degree	98	(127/130)	95	(127/134)	96
det	98	(177/181)	96	(177/185)	97
det_type	98	(178/181)	97	(178/183)	98
fragment	0	(0/0)	0	(0/2)	0
fut	75	(3/4)	100	(3/3)	86
gend	100	(441/442)	98	(441/449)	99
gl	100	(17/17)	71	(17/24)	83
gr	77	(36/47)	95	(36/38)	85
measured	100	(1/1)	100	(1/1)	100
mo	89	(321/359)	84	(321/383)	87
mod	100	(4/4)	100	(4/4)	100
mood	97	(128/132)	99	(128/129)	98
name_mod	89	(24/27)	89	(24/27)	89
num	99	(632/638)	99	(632/639)	99
number	100	(15/15)	83	(15/18)	91
oa	92	(45/49)	90	(45/50)	91
obj	96	(177/184)	94	(177/189)	95
oc_fin	82	(9/11)	82	(9/11)	82
oc_inf	83	(19/23)	83	(19/23)	83
op	88	(14/16)	88	(14/16)	88
pass_asp	81	(13/16)	93	(13/14)	87
pd	87	(27/31)	90	(27/30)	89
perf	100	(8/8)	80	(8/10)	89
pers	98	(173/177)	99	(173/174)	99
pron_form	100	(7/7)	100	(7/7)	100
pron_type	94	(85/90)	98	(85/87)	96
quant	86	(19/22)	100	(19/19)	93
rc	62	(5/8)	45	(5/11)	53
sb	93	(182/196)	92	(182/197)	93
tense	97	(128/132)	99	(128/129)	98
tiger_id	90	(102/113)	93	(102/110)	91
total	**95.5**		**94.6**		**95.0**

Table 9.9: Results for automatic F-structure annotation on gold trees (TUBA100 gold standard)

Chapter 10

Parsing

10.1 Introduction

This chapter presents different approaches to automatic treebank-based grammar extraction (related to the representation of crossing branches in TiGer), parsing and evaluation for German, based on the TiGer and TüBa-D/Z treebanks. First I describe the research methodology used in my work, which aims at comparing the quality of different architectures based on the two treebanks (Section 10.2) as well as comparing the influence of different conversion methods to transform the non-projective TiGer dependency graphs into CFG representations (Section 10.2.1).

I present parsing experiments using automatically F-structure-annotated resources based on the two German treebanks, adapted to different feature sets (TiGer DB, DCU250 and TUBA100) (Figure 10.1). First I evaluate the performance of different parsers and architectures based on the TiGer treebank on the c-structure and F-structure level against the TiGer DB gold standard (Section 10.3.2) and the DCU250 (Section 10.3.3). In Sections 10.3.3.1 and 10.3.3.2 I provide an error analysis and discuss problems specific to different settings in the grammar extraction architecture, mainly concerning different approaches to the assignment of grammatical function labels in parse trees and their impact on F-structure results. I compare two methods: (i) the assignment of grammatical function labels by the Berkeley parser Petrov and Klein (2007) and (ii) by an SVM-based grammatical function labeller (FunTag) (Chrupala et al., 2007).

In Section 10.3.4 I report c-structure and F-structure results for TüBa-D/Z-trained parsing resources. The evaluation against the hand-crafted gold standards is complemented by a CCG-style evaluation (Hockenmaier and Steedman, 2002a) against a larger test set of 2000 automatically F-structure-annotated gold trees from each the TiGer treebank, and from the TüBa-D/Z. Section 10.3.6 discusses the main differences between the grammar extraction architectures based on the two different treebanks, TiGer and TüBa-D/Z. In Section 10.4 I summarise my main findings.

10.2 Approaches to Treebank-Based Grammar Extraction, Parsing and Evaluation

The two treebanks and the five gold standard resources described above support different approaches to grammar extraction, F-structure annotation and evaluation for parsing (Figure 10.1). My general approach is as follows: I follow the *pipeline parsing architecture* (Figure 7.3) and extract a PCFG from each treebank. For TiGer, I have to resolve the crossing branches in the trees in a preprocessing step. I test two different approaches to tree conversion: (i) the split-node conversion of Boyd (2007) and (ii) the raised-node conversion, as described in Kübler (2005).

10.2.1 Raised versus Split - What's the Difference?

The TiGer treebank uses trees with crossing branches to represent non-local (or non-projective) dependencies. Trees with crossing branches cannot be processed by standard state-of-the-art data-driven and CFG-based parsing technologies. Because of this, trees with crossing branches have to be transformed into trees without crossing branches in a preprocessing step, prior to grammar acquisition or parser training. The standard technique for doing this is outlined in Kübler (2005). Her method works by attaching all non-head child nodes in a discontiguous tree structure higher up in the tree, until all crossing branches have been resolved (Figures 10.2,10.3). This approach has the disadvantage of breaking up the original tree structure and introducing inconsistencies in the trees, which compounds the problem of learnability for the flat annotation in the TiGer treebank,

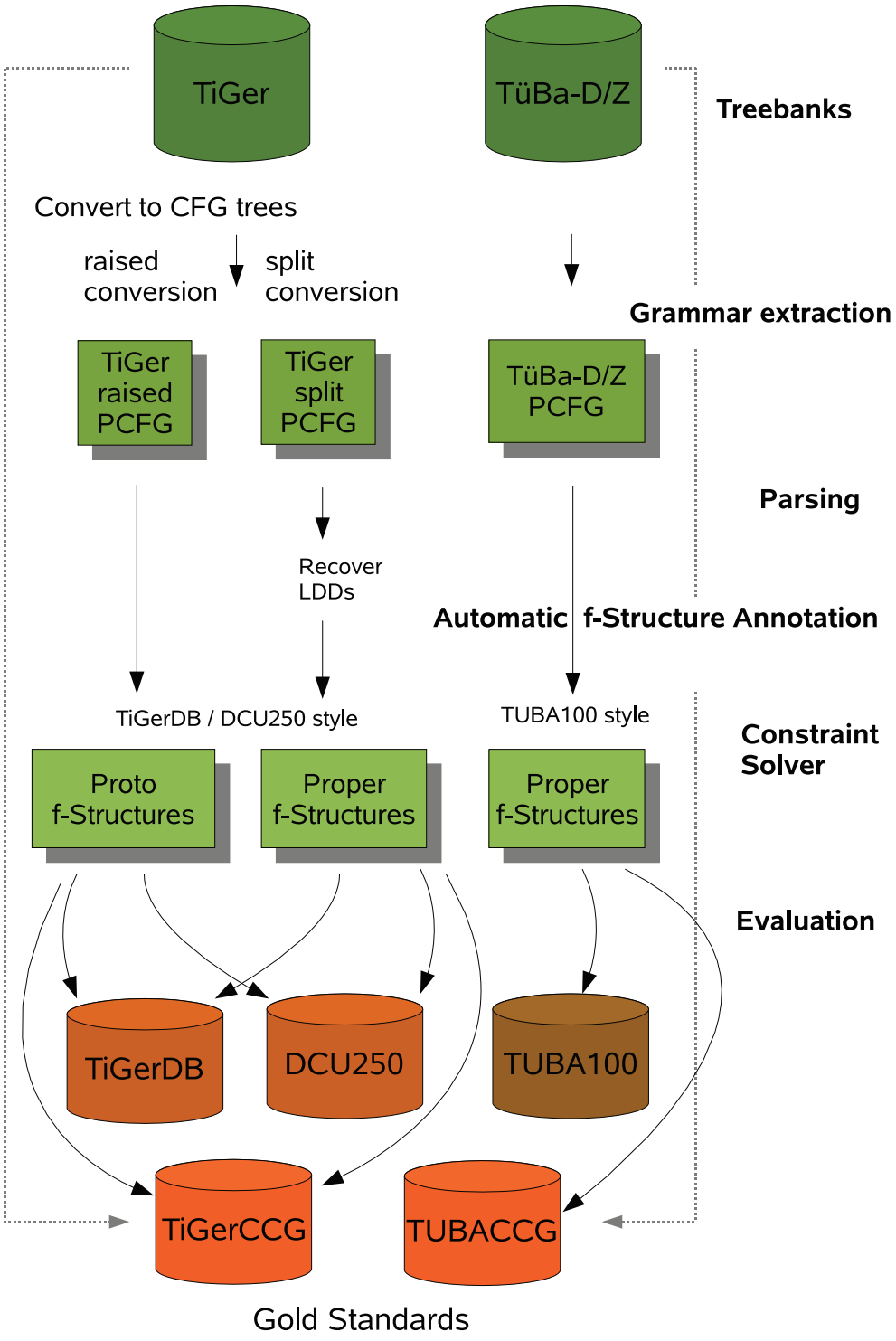


Figure 10.1: Different approaches to grammar extraction, f-stucture annotation and evaluation for parsing

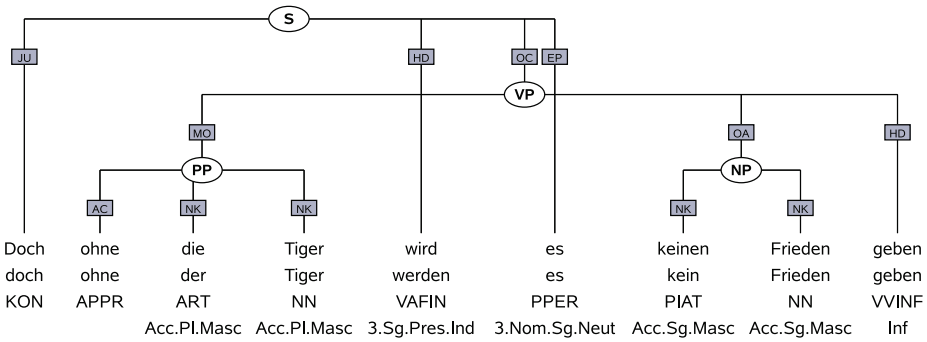


Figure 10.2: Conversion of crossing branches into CFG trees: original tree

resulting in a high number of long, low-frequency rules.

Figure 10.2 shows a TiGer tree with crossing branches for the sentence in (25) from the TiGer treebank.

(25) Doch ohne die Tiger wird es keinen Frieden geben.
but without the tigers will it no peace give.
"But without the tigers there will be no peace."

Figure 10.3 displays the same tree with crossing branches resolved, using Kübler's raised-node technique. In the original TiGer tree the PP *(ohne die Tiger)* and the NP *(keinen Frieden)* are both child nodes of the discontinuous VP. In the raised-node conversion the information about the original attachment of the PP is lost, and so is the information that the PP is a verb modifier of *geben* (to give).

Boyd (2007) proposes an improved method for resolving crossing branches in TiGer by annotating partial nodes in the trees. This method allows us to encode the original dependency relations in the converted tree and to reconstruct the original tree after parsing. In Boyd's split-node conversion of the tree in Figure 10.2, the original annotation is encoded by newly inserted paired split nodes, which are marked by an asterisk (Figure 10.4). This encoding preserves the information that the PP is a child of the VP by attaching it to a "partial"

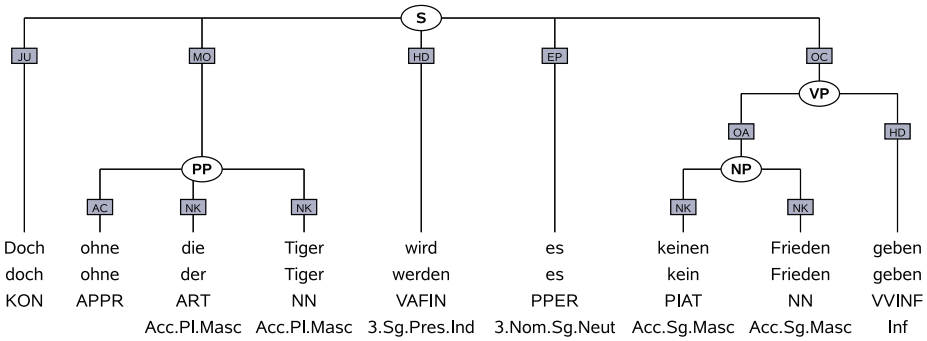


Figure 10.3: Conversion of crossing branches into CFG trees: raised-node (Kübler, 2005)

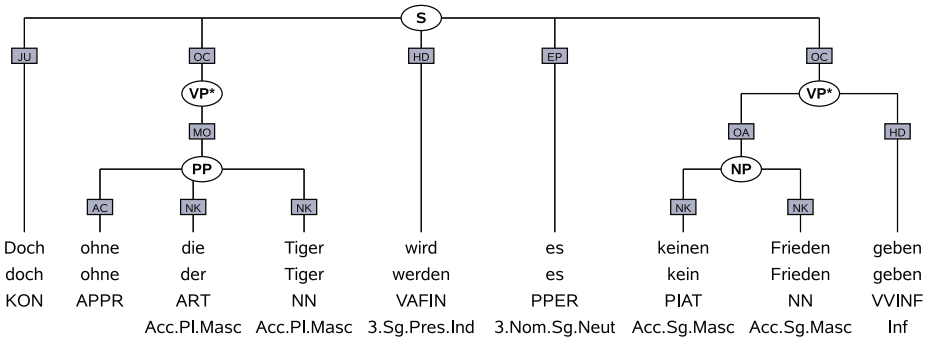


Figure 10.4: Conversion of crossing branches into CFG trees: split-node (Boyd, 2007)

VP node in the converted tree. After parsing the partial nodes can be merged again and the original tree structure can be recovered, provided that the parser correctly assigned the paired partial nodes in the parser output.

After converting the trees we have two versions of the TiGer treebank (raised-node and split-node). The raised-node conversion results in a lossy version of TiGer, while the split-node conversion still encodes the original non-local dependencies in the trees. In TüBa-D/Z, Non-Local Dependencies (NLDs) are encoded by the means of grammatical function labels. As a result, two of the extracted PCFGs underpinning the treebank-based pipeline LFG parsing architecture (Figure 10.1) are "deep" (TiGer split-node and TüBa-D/Z), while the third (TiGer raised-node) is a "shallow" grammar which can not reproduce the original non-local information in the training data in the parser output.

10.2.2 Automatic F-structure Annotation

After extracting the CFG grammars I use the three PCFGs to parse the test sets. The extracted PCFGs include grammatical function labels, merged with the node labels. I re-convert the parser output of the TiGer split-node PCFG into discontiguous graphs. In the next processing step in the pipeline parsing architecture I automatically annotate the parser output trees with LFG F-structures.

The different gold standards offer the following possibilities with regard to F-structure annotation: we can annotate the original TiGer treebank trees with TiGer DB-style grammatical functions and use the TiGer DB for evaluation, or we can annotate the trees with DCU250-style grammatical functions and evaluate the resulting F-structures against the DCU250. For the TüBa-D/Z we annotate the original treebank trees with TUBA100-style grammatical functions, which can be evaluated against the TUBA100 gold standard.

The results of the annotation process are F-structure-annotated parse trees, either in the style of the TiGer DB, the DCU250 or the TUBA100. The functional equations in the trees are collected and passed over to the constraint solver, which produces F-structures. From the TiGer raised-node parse trees we obtain "proto" F-structures with long-distance dependencies unresolved. The TiGer split-node parse trees as well as the TüBa-D/Z parser output allow for the generation of proper F-structures as information about non-local dependencies is encoded in

the tree. The resulting F-structures are evaluated against the TiGer DB, the DCU250 or the TUBA100, depending on the set of grammatical functions used in the annotation.

There is yet another possible approach to the evaluation of the automatically generated F-structures. In the first step the original trees from the TiGer treebank are annotated with F-structure equations. The annotated gold trees can be used to *automatically* create a dependency gold standard for the evaluation of the F-structures obtained from raw text (CCG style evaluation, (Hockenmaier and Steedman, 2002a)). The original trees from the two treebanks represent long-distance dependencies, so the resulting F-structures are proper F-structures with LDDs resolved. This allows me to produce large data sets for the evaluation of F-structures either in the TiGer DB-style, TUBA100-style, or DCU250-style (referred to as TiGerCCG and TUBACCG in Figure 10.1).

10.3 Parsing into LFG F-structures

In Chapter 9 I showed that the improved LFG F-structure annotation algorithm for German produces good results when annotating gold treebank trees. Now I want to investigate whether the results of our method are still respectable when applied to parser output trees. My German AA strongly relies on the grammatical function labels present in the treebank trees. In contrast to English, configurational information does not provide much help when disambiguating the functional structure in a German sentence. Instead, my approach relies on the combined information provided by syntactic categories, function labels and contextual information in the treebank trees. Therefore it is to be expected that when parsing with combined syntactic category and grammatical function label information, parser errors will have a strong impact on the quality of the generated F-structures.

Results in the recent shared task on parsing German (Kübler, 2008) overall are quite discouraging. The best contribution was made by the Berkeley parser (Petrov and Klein, 2008), which achieved a precision of 69.2 and a recall of 70.4% (`evalb`, syntactic categories + grammatical functions) when trained on the TiGer treebank, using gold part-of-speech tags (including gold grammatical function labels for terminal nodes). This means that in current state-of-the-art treebank-

based parsing for German around 30% of the node labels assigned by the parser are incorrect, which (for this architecture where the parser learns the function labels) suggests an upper bound for the task of treebank-based LFG parsing for German. In this context, I investigate the following research questions:

- What is the impact of different treebank designs on treebank-based grammar acquisition?
- Which architecture for grammar acquisition is better suited for German?
- What is the upper bound for treebank-based grammar acquisition for German, based on erroneous parser output trees? What are the main problems, and which strategies can help to overcome these problems?

To enable a meaningful comparison of the two German treebanks, training sets of the same size from TiGer and TüBa-D/Z are required. Therefore I removed all gold standard sentences from the two treebanks and extracted a training set with 25,000 sentences from each of the treebanks.

The training sets were created as follows: I divided the two treebanks into 27 parts, using 27 "buckets". I put the first sentence into the first bucket, the second into bucket 2, and so on. After reaching the 27th bucket, I started again with the first one. For the TüBa-D/Z this results in 27 buckets with 1000 sentences each (I removed the TUBA100 gold standard sentences as well as the remaining 25 sentences). Then I combined the first 25 buckets into a training set with 25,000 sentences and put all sentences from bucket 26 and 27 into a test set for the CCG-style evaluation. For TiGer I proceeded in a similar way, but stopped after all buckets were filled with 1000 sentences each. The remaining treebank sentences have been discarded.

In order to investigate the impact of the size of the training set on the quality of the F-structures, I also created a second training set for TiGer. The large training set consists of all sentences in the TiGer treebank except sentences 8000-1000 (which include the TiGer DB and the DCU250). The exact size of the large training set is 48,473 sentences.

10.3.1 Experimental Setup

In the experiments I used three different parsers: BitPar (Schmid, 2004), the Stanford Parser (Klein and Manning, 2003) and the Berkeley Parser (Petrov and Klein, 2007). The Berkeley Parser is a purely data-driven parser, using a split-and-merge technique to automatically refine the training data. The splits result in more and more fine-grained subcategories, which are merged again if not proven useful. The model is language-agnostic and achieved best results in the shared task on parsing German at ACL 2008 (Petrov and Klein, 2008).

All three parsers were trained on the TiGer and TüBa-D/Z training sets (25,000 trees) and on the large TiGer training set (48,473 trees). For BitPar and the Stanford Parser we included grammatical functions in the treebank by merging the edge labels with the categorial node labels. As a result we get a much larger set of node labels for the parsers to learn (approximately 720 node labels for TiGer and 360 for the TüBa-D/Z). The larger number of different node labels for TiGer is due to the flat annotation scheme in the TiGer treebank, which results in terminal nodes being assigned many different grammatical function labels like subject (SB), accusative object (OA), dative object (DA), and so on. In TüBa-D/Z, due to the more hierarchical tree structure and the annotation of unary nodes, terminal nodes are assigned two different grammatical function labels only: head (HD) and non-head (-). For the Berkeley parser I report results for three different settings:

1. grammatical functions learned by the parser (berk.par)

2. parser trained on treebank trees without grammatical function labels and grammatical functions added in a post-processing step by an SVM-based grammatical function labeller (FunTag, (Chrupala et al., 2007)), trained on gold treebank trees (berk.fun)[20]

3. same as (2) but grammatical functions added in a post-processing step by the SVM-based function labeller, trained on parser output (berk.fun.par)

The first setting is the same as for BitPar and the Stanford Parser, where I merged grammatical function labels and syntactic node labels into new, atomic

[20] I am grateful to Grzegorz Chrupala who provided the grammatical function labelling software.

labels. In the second setting I removed all grammatical functions from the treebank and trained the Berkeley parser on syntactic categories only. After parsing I applied the automatic grammatical function labeller to the parser output trees. The function labeller then assigns grammatical function labels to the syntactic nodes in the trees (two-step architecture).

FunTag treats the function labelling problem as a binary classification task. For each syntactic node in the tree, FunTag extracts a set of features from the gold trees, capturing categorial, configurational and lexical information about the node and its context. Each training example is assigned a class label (a specific grammatical function or NULL, if the particular node is not associated with this specific grammatical function). Off-the-shelf SVM software[21] is trained on the feature set extracted from the gold trees (berk.fun) or parser output trees (berk.fun.par).

Machine learning-based classifiers yield best results on data sets which are as similar as possible to the training instances. As we want to assign grammatical function tags to parser output trees, it seems reasonable to train the classifier on parser output trees instead of gold trees. Chrupala et al. (2007) tested this training method on re-parsed data from the English Penn-II treebank and achieved a significant improvement for the function labelling task over training on the original treebank trees.

I used the Berkeley parser to re-parse the TiGer treebank and applied the improved training method outlined in Chrupala et al. (2007) to the re-parsed treebank (berk.fun.par). All c-structure parsing results are evaluated with `evalb` and report labelled F-scores for sentences with sentence length <= 40 without grammatical functions (noGF) and with grammatical functions (GF).[22] All TiGer results reported in Section 10.3 are for "shallow" parsers trained on the raised-node conversion of the TiGer treebank. Results for "deep" parsers trained on the split-node converted TiGer treebank are discussed in Section 11.2.

[21] SVM_{light} (Joachims, 2002)

[22] Restricting c-structure evaluation to shorter sentences allows a more meaningful comparison with related work, where `evalb` results are usually reported for sentences with length <= 40. Results for F-structure evaluation in my experiments consider sentences of all lengths.

10.3.2 C-Structure and F-Structure Parsing Results for the TiGer DB

Table 10.1 presents c-structure and F-structure parsing results for the three different parsers trained on the TiGer treebank, generating TiGerDB-style LFG F-structures. For both c-structure and F-structure evaluation, I report coverage: on c-structure level the number of sentences receiving a parse tree, and on F-structure level the percentage of sentences for which the constraint solver produces an F-structure, resulting in a set of F-structure dependency triples for the parse tree.

For training on 25,000 trees for c-structure results there is a large difference of around 10% between F-scores for the different parsers. BitPar achieves an F-score of 70.9% (noGF) and 60.1% (GF) and is clearly outperformed by the other two parsers (stanford: 74.5 (noGF) and 63.2 (GF), berk.par: 79.3 (noGF) and 70.2 (GF)).

The Berkeley parser trained on syntactic categories without grammatical functions (berk.fun) produces the best c-structure results (excluding GFs from the evaluation) for the TiGer treebank with an 81.0% F-score. After applying the FunTag grammatical function labelling software trained on gold trees, we achieve an `evalb` F-score of 70.9% (GF, berk.fun), which is slightly higher than the one for the parser-assigned grammatical functions (70.2% (GF, berk.par)). The results for the function labeller trained on parser output, however, are slightly worse than for the setting where we train the labeller on gold treebank trees (GF, berk.fun: 70.9; GF, berk.fun.par: 70.8).

Not surprisingly, for all three parsers (bitpar, stanford, berk.par) parsing results improve when training on the larger TiGer training set (>48,000 trees) (Figure 10.2). For the parsers trained on syntactic node labels + grammatical function (bitpar, stanford, berk.par), we observe an improvement in F-score of 2.6% for BitPar and the Berkeley parser (noGF) and of 3.1% for the Stanford parser (noGF), while for the Berkeley parser trained on syntactic nodes only (berk.fun, berk.fun.par) the improvement is somewhat smaller with 2.2% (noGF).

Including the grammatical function labels in the evaluation (GF), we observe the same general trend: the Stanford parser makes the most of the larger training set and shows an improvement of 3.4%, followed by the Berkeley parser with 3.1%

	bitpar	stanford	berk.par	berk.fun	berk.fun.par
TIGER25000 - c-structure evaluation					
length <= 40	1762	1762	1762	1762	1762
# parse	1752	1759	1757	1759	1759
F-score noGF	70.9	74.5	79.3	81.0	81.0
F-score GF	60.1	63.2	70.2	70.9	70.8
tagging acc.	94.8	97.2	96.0	97.0	97.0
F-structure evaluation - development set					
# sent	1366	1366	1366	1366	1366
% f-struct.	87.8	92.9	89.3	92.8	90.3
Precision	70.4	73.9	75.9	77.1	78.3
Recall	71.8	74.1	76.6	64.7	62.1
F-score	71.1	74.0	76.2	70.3	69.3
F-structure evaluation - test set					
# sent	500	500	500	500	500
% f-struct.	85.6	89.2	85.4	90.6	88.4
Precision	66.7	70.9	73.1	75.2	75.4
Recall	67.7	70.1	73.7	58.3	55.4
F-score	67.3	70.5	73.4	65.7	63.9
TIGER48000 - c-structure evaluation					
# parses	1759	1758	1757	1759	1759
F-score noGF	73.5	77.6	81.9	83.2	83.2
F-score GF	62.6	66.6	73.3	73.0	70.4
tagging acc.	96.1	97.8	97.4	98.0	98.0
F-structure evaluation - development set					
# sent	1366	1366	1366	1366	1366
% f-struct.	87.5	93.3	91.4	94.0	90.5
Precision	72.3	75.2	76.9	78.4	77.9
Recall	74.1	75.0	77.7	66.3	62.7
F-score	73.2	75.1	77.3	71.9	69.5
F-structure evaluation - test set					
# sent	500	500	500	500	500
% f-struct.	85.4	87.8	88.0	90.0	90.6
Precision	69.1	72.5	74.8	75.6	75.3
Recall	70.2	72.0	74.8	60.3	54.4
F-score	69.7	72.2	74.8	67.1	63.1

Table 10.1: C-structure parsing results (labelled F-score) and F-structure evaluation for different German grammars and parser (TiGer DB)

	bitpar	stanford	berk.par	berk.fun	berk.fun.par
TIGER25000 - c-structure evaluation					
F-score noGF	70.9	74.5	79.3	81.0	81.0
F-score GF	60.1	63.2	70.2	70.9	70.8
tagging acc.	94.8	97.2	96.0	97.0	97.0
TIGER48000 - c-structure evaluation					
F-score noGF	73.5	77.6	81.9	83.2	83.2
F-score GF	62.6	66.6	73.3	73.0	70.4
tagging acc.	96.1	97.8	97.4	98.0	98.0

Table 10.2: C-structure parsing results (labelled F-score) for different German grammars and parser (TiGer DB) for training sets with 25,000 and 48,000 trees

and BitPar with 2.5%. For the Berkeley and Stanford parsers the improvement on the larger amount of training data is more profound when including grammatical function labels in the evaluation. This confirms our suspicion that merging syntactic nodes with grammatical function labels increases the problem of sparse data for the TiGer treebank. In the two-step architecture, where grammatical function labels are assigned by FunTag, we do not observe the same increase in results. Training the grammatical function labeller on gold treebank trees (berk.fun), F-score increases 2.1% (GF), while for the berk.fun.par setting, where I trained FunTag on parser output trees, there is a decrease in F-score of 0.4%.

Figure 10.5 shows the learning curve for the Berkeley parser trained without grammatical functions. In the beginning, the curve is very steep up to a training size of around 20,000 trees. After that, adding more training data does not have such a strong effect on F-scores any more, and from a training size of 35,000 on there is a slight improvement only, if any. It seems as if the problem of parsing German is unlikely to be solved by merely increasing the size of the treebanks.

For the Berkeley parser trained on a combination of syntactic nodes and grammatical functions, the number of labels to be learned by the parser increases dramatically. This is likely to result in data sparseness, and I expect a learning effect even at a training size of more than 40,000 trees. Figure 10.6 shows the learning curve for the Berkeley parser when trained on the merged node labels including grammatical functions. There is a profound learning effect resulting in a very steep rise for the first 27,500 trees in the training set. From then on the curve does not flatten, but takes a jagged course. We achieve best results for the maximum training size of 48,473 trees. Extrapolating from this it is likely that

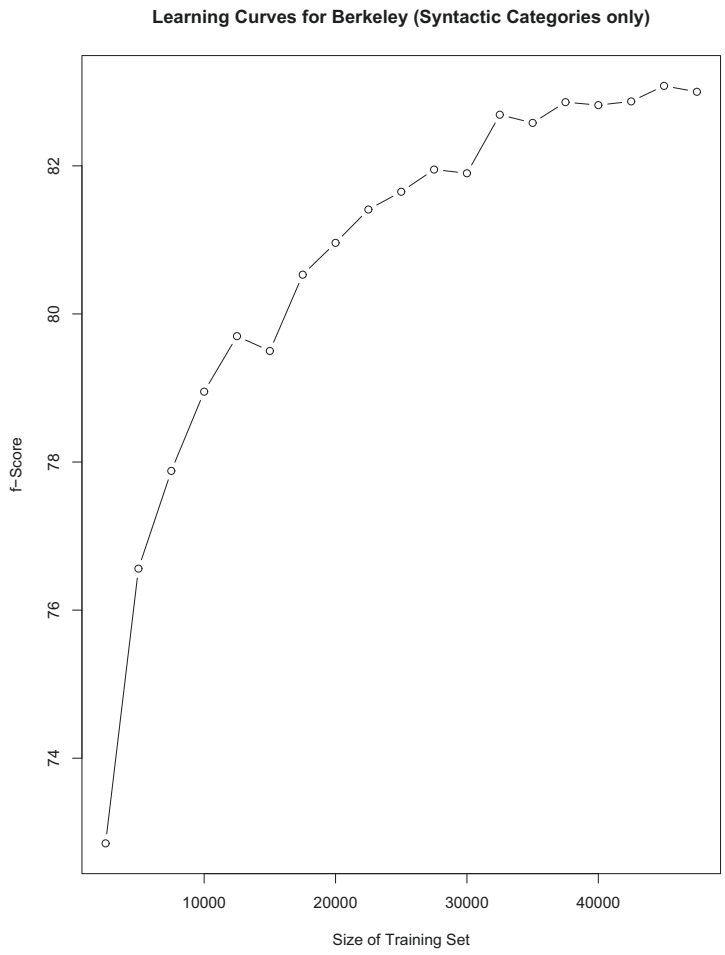


Figure 10.5: Constituency parsing learning curves for the Berkeley parser (no GF, berk.fun)

adding more training data would succeed in further boosting parser performance for the Berkeley parser trained on grammatical functions.

Most interestingly, Berkeley constituency parsing F-scores are significantly better when trained on syntactic nodes only (79.3 vs. 81.0 (noGF) for TIGER25000 and 81.9 vs. 83.2 (noGF) for TIGER48000). However, one should keep in mind that parse trees without grammatical functions do not give a sufficient representation of syntactic information in German, as they fail to encode basic information about predicate-argument structure.

For F-structure evaluation (Table 10.1) we observe the same parser ranking as for the constituent-based evaluation. For both the development and test set, the Stanford parser gives better results than BitPar, and the Berkeley parser trained on a combination of syntactic nodes and grammatical functions outperforms the Stanford parser. When trained on syntactic nodes only (berk.fun, berk.fun.par), performance for the F-structures generated from the Berkeley parser output decreases drastically. While precision is higher with around 78% for the development set and close to 76% for the test set, recall is between 15-20% lower than for the berk.par F-structures. Despite achieving very similar `evalb` results for the setting including grammatical functions, it seems as if there is a fundamental difference between berk.par and berk.fun parse trees. This is a surprising finding which I investigate and discuss in Section 10.3.3.2. While showing low recall, the parser output for the combination of Berkeley parser and function labeller (berk.fun and berk.fun.par) yields the highest number of F-structures. This seems to be somewhat contradictory, but simply means that berk.fun and berk.fun.par produce a higher number of F-structures than the other parsers, while the F-structures themselves are not complete. Take for example a parse tree with a subordinated clause where FunTag failed to assign a grammatical function label to the subclause. As a result, the subclause may not be represented on F-structure level, causing a severe decrease in recall.

10.3.3 C-Structure and F-Structure Parsing Results for the DCU250

Table 10.3 presents c-structure and F-structure parsing results for the three different parsers trained on the TiGer treebank, generating DCU250-style LFG F-

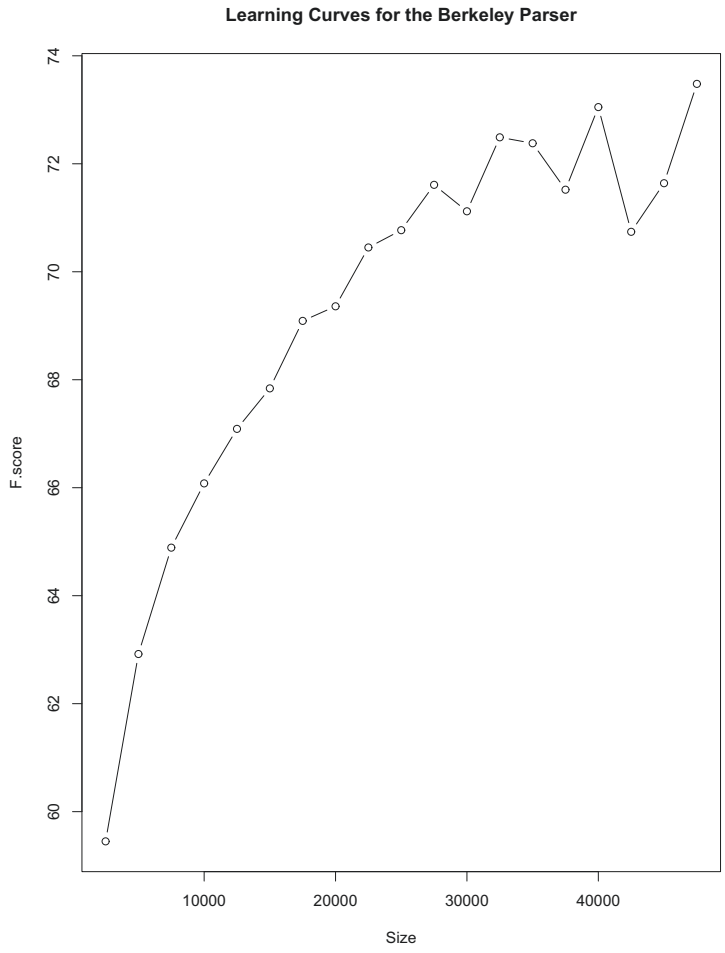


Figure 10.6: Constituency parsing learning curves for the Berkeley parser (GF, berk.par)

structures.

Similar to the TiGerDB-style F-structures, there is a gap of around 10% between `evalb` F-scores for the different parsers. BitPar produces an F-score of 70.1% (noGF) and 58.6% (GF), the Stanford parser achieves 73.7% (noGF) and 62.2% (GF), and the Berkeley parser gives results between 79.3%-81% (noGF: berk.par, berk.fun, berk.fun.par) and around 70% when including grammatical functions in the evaluation (GF). `Evalb` results for the DCU250 for all parsers are slightly lower than results for the TiGer DB.

F-structure results show the same trend as for the TiGerDB, but on average are approximately 10% higher. As before, the two-step architecture (Berkeley/FunTag) produces F-structures with highest precision, but at the cost of a severe decrease in recall. Somewhat unexpected are the higher results on the test set for the DCU250 for most parsers and settings. Only the Berkeley parser trained on syntactic nodes + grammatical functions (berk.par) produces better results for the DCU250 development set than for the test set. A possible explanation are the high percentage of sentences (12%) in the development set which did not receive a valid F-structure. Because of this it is likely that more difficult sentences have been excluded from the evaluation.

The better results for the test set suggest that the development set is somewhat harder to parse than the test set. This assumption is supported by the differences in sentence length in both data sets. In the test set the average sentence length is 22.1 with only 3 sentences showing a word length > 40, while in the development set the average sentence length is 23.8, including 13 sentences with more than 40 words. The longest sentence in the test set has a sentence length of 49 words, while in the development set there are 5 sentences with more than 60 words, and the maximum sentence length is 100 words.

Overall, best results for the DCU250 are achieved by the Berkeley parser for the parsing model including grammatical functions in the node labels (berk.par). For the 25,000 training set we get an F-score of 80.5% on the test set, and for the large training set (48,000 sentences) it further increases up to 83.0%. This, however, comes at the cost of a high number of sentences not receiving a F-structure. Precision for the two-step architecture (Berkeley/FunTag) is close to 90% (TIGER48000), but achieves low recall only, while the number of F-structure clashes for the berk.par setting is higher than for berk.fun and berk.fun.par.

	bitpar	stanford	berk.par	berk.fun	berk.fun.par
TIGER25000 - c-structure evaluation					
# sent < 40	234	234	234	234	234
# parse	233	234	234	234	234
F-score noGF	70.1	73.7	76.6	79.3	79.3
F-score GF	58.6	62.2	66.9	68.4	68.0
tagging acc.	94.6	96.6	95.4	96.5	96.5
TiGer F-structure evaluation - development set					
# sent	125	125	125	125	125
% f-struct.	87.2	91.2	88.8	92.2	90.4
Precision	76.5	79.6	81.0	86.7	86.7
Recall	76.2	74.5	80.7	58.0	57.8
F-score	76.3	77.0	80.8	69.5	69.4
TiGer F-structure evaluation - test set					
# sent	125	125	125	125	125
% f-struct.	90.4	95.2	92.0	93.6	93.6
Precision	77.0	80.9	81.4	86.7	86.5
Recall	77.7	79.9	79.7	68.1	68.4
F-score	77.3	80.4	80.5	76.3	76.4
TIGER48000 - c-structure evaluation					
# sent < 40	234	234	234	234	234
# parses	234	234	226	234	234
F-score noGF	71.6	75.2	81.9	81.4	81.4
F-score GF	59.6	63.8	72.4	70.8	70.9
tagging acc.	96.0	97.6	96.9	97.6	97.6
TiGer F-structure evaluation - development set					
# sent	125	125	125	125	125
% f-struct.	88.8	95.2	88.8	92.0	88.0
Precision	77.1	80.0	84.7	89.3	89.3
Recall	77.2	75.8	83.7	63.3	62.0
F-score	77.1	77.9	84.2	74.1	73.2
F-structure evaluation - test set					
# sent	125	125	125	125	125
% f-struct.	88.8	96.8	94.4	96.8	96.0
Precision	78.1	81.7	83.6	86.8	87.3
Recall	79.0	80.0	82.5	70.0	69.7
F-score	78.5	80.8	83.0	77.5	77.5

Table 10.3: C-structure parsing results (labelled F-score) and F-structure evaluation for different German grammars and parser (DCU250)

	berk.par	**berk.fun**
# F-structures	1220	1268
# clashes	146	98
error type: ≥ 2 *GF in local tree*		
HD	46	22
OA	29	18
SB	23	26
OC	19	5
DA	3	1
all with ≥ 2 GF	120	72

Table 10.4: Types of errors in berk.par and berk.fun

10.3.3.1 Error Analysis

The observations from the TiGer DB/DCU250 evaluation raise the following questions:

1. What causes the higher number of clashes resulting in fewer F-structures in the Berkley parser output when trained on syntactic nodes + grammatical functions (berk.par)?

2. What is the reason for the low recall for F-structures generated on the output of the function labeller?

To answer the first question I looked at the parse trees in the TiGer DB development set which did not receive a F-structure. For the Berkeley parser trained on categories and grammatical functions (berk.par), there are 146 F-structure clashes, while for the FunTag-labeled trees from the Berkeley parser trained on syntactic nodes, only (berk.fun) 98 trees did not receive an F-structure (Table 10.4). 41 of the trees exhibiting a clash were the same in both settings, berk.par and berk.fun.

For the 146 trees in the berk.par output not receiving a F-structure, most clashes (120) were caused by the parser assigning the same governable grammatical function twice to child nodes of the same parent node, thus violating the LFG coherence condition. 46 out of the 146 trees had an S or VP node with two heads

(HD), 23 had more than one subject (SB), 29 had more than one accusative object (OA), 3 two dative objects (DA), and 19 more than one clausal object (OC) child node.

For the 98 trees in the berk.fun output not receiving a F-structure, 22 out of the 98 trees had a clause with two heads (HD), 26 had more than one subject (SB), 18 had more than one accusative object (OA), 1 two dative objects (DA), and 5 more than one clausal object (OC) child node. This shows that most of the clashes, namely 120 out of the 146 clashes in the berk.par parser output and 72 out of the 98 clashes in the berk.fun output are caused by the assignment of 2 or more identical GF labels in the same local tree.

This type of error is caused by the split-and-merge technique applied by the Berkeley parser and by horizontal Markovisation, where long grammar rules are broken up to avoid data sparseness. Hence the parser does not have as much context information as before, which results in errors like the ones described above. FunTag has a similar problem: the grammatical function labelling task is designed as a binary classification problem, where each node in the tree is assigned a GF label, independently of the other node labels in the tree.

Another reason for the high number of clashes is POS tag errors. In many cases where there are two head child nodes in a sentence or verb phrase, the parser assigned the label VVFIN (finite full verb) to an infinite verb or a past participle (Figure 10.7). In the output of the Berkeley parser trained on syntactic node labels only, these POS errors do not occur. The problem arises from the flat annotation in the TiGer treebank, where many terminal nodes are directly attached to the sentence node, with grammatical function labels attached to the terminals. This blows up the set of POS tags when merging grammatical function labels with the node labels. As a result it becomes much harder for the parser to assign the correct POS tag when trained on the larger label set of syntactic nodes + grammatical functions.

Coordinations constitute another problem. Figure 10.8 shows a Berkeley parse tree where the parser did not recognise the coordinated sentence, but attached all terminal nodes to the same S node. As a result the tree shows a very flat structure with two finite verbs directly attached to the sentence node. As both finite verbs are assigned the label (HD) by the Parser or FunTag, respectively, parse trees with this particular error do no get a F-structure. This error type

(26) [...] darüber läßt sich trefflich streiten
[...] about it let refl felicitous dispute
"that's open to dispute"

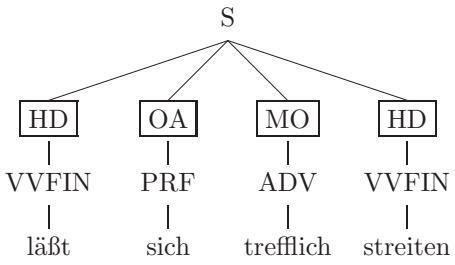


Figure 10.7: POS tag error by the Berkeley parser trained with GF

occurs for both the parser-assigned as well as the FunTag-assigned grammatical functions.

The error analysis above explains the lower number of trees without a valid F-structure in the berk.par parser output, but does not account for the low recall values for the two-step labelling architecture based on the Berkeley parser trained on syntactic nodes only and FunTag. The next section takes a detailed look at the output of the grammatical function labeller.

10.3.3.2 Evaluating FunTag

Despite the better constituent-based parsing results (`evalb`, GF (berk.par, berk.-fun, berk.fun.par)), results for F-structure evaluation are better for F-structures generated from Berkeley parser output when trained on extended node labels including grammatical functions (berk.par) than for the two-stage function labelling architecture. This is more evidence for the already strong claim that PARSEVAL scores do not reflect real parser output quality.

In fact, there may be a structural difference between parser output trees from parsers trained on a combination of syntactic nodes and grammatical functions and parsers trained on syntactic nodes only, which is not reflected in the PARSEVAL results. Parse trees generated by a grammatical function-trained parser might be better at capturing important properties of the semi-free German word order, even if this is not reflected in the `evalb` evaluation. To investigate the differences between the different types of parse trees I first evaluate the sets of

(27) Boernsen dementiert dies zwar energisch, streitet aber ein Interesse an
Boernsen denies this indeed energetically, disclaims but an interest in
dem Job nicht grundsätzlich ab
the job not generally particle
"Boernsen resolutely denies this, but does not generally deny an interest in the job"

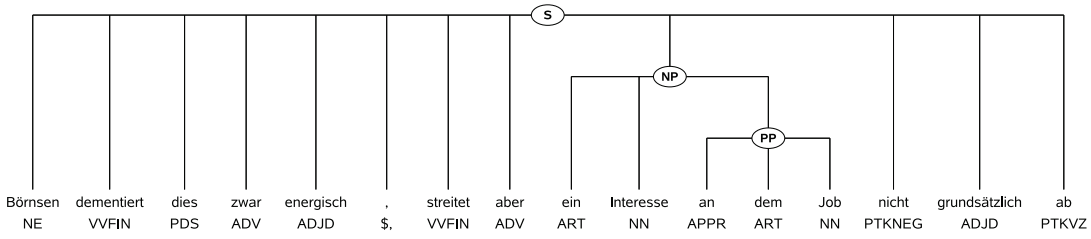


Figure 10.8: Berkeley parser error

grammatical functions a) learned by the parser and b) assigned by the FunTag function labelling software. Then I present an error analysis for sentences which did not obtain a F-structure.

In the evaluation I compare results for TiGer treebank grammatical functions assigned by the parser (bitpar, stan, berk.par) and by FunTag (berk.fun, berk.fun.par). In the berk.fun setting the function labelling software was trained on gold trees from the TiGer treebank (TIGER48000). In the berk.fun.par setting, the SVM is not trained on gold treebank trees, but on training instances extracted from parser output trees. Table 10.5 shows F-scores for grammatical function labels for the TiGer DB test set. I also assigned grammatical functions to gold treebank trees (gold) using FunTag, which yields an upper bound of 97% F-score. Overall results for the two-step approach, where grammatical function tags are assigned by FunTag after parsing, are slightly higher than for the parser-assigned GFs. For FunTag trained on gold treebank trees we obtain an F-score of 86.2%, while the improved training method (training on parser output trees) achieves best results with 86.8%. Results for parser-assigned grammatical function tags are lower with 78.4% (BitPar), 81.7% (Stanford) and 84.6% (Berkeley).

This is a bit of a puzzle: results for the CFG trees as well as for the grammati-

GF	**bitpar**	**stan**	**berk.par**	**berk.fun**	**berk.fun.par**
AC	97.6	98.9	98.7	99.0	99.0
ADC	-	-	-	-	-
AG	62.0	72.1	77.7	75.0	75.8
AMS	35.3	58.1	53.8	42.9	44.4
APP	36.9	43.9	52.8	58.0	59.5
AVC	20.0	72.7	75.0	70.6	85.7
CC	52.6	62.0	56.3	43.1	43.0
CD	94.6	96.7	96.7	96.4	97.1
CJ	57.9	58.3	65.1	72.2	72.6
CM	72.4	83.6	77.4	77.9	77.9
CP	92.2	96.7	95.9	97.1	97.1
CVC	3.6	10.9	20.0	60.4	63.0
DA	12.6	28.8	45.1	50.5	50.5
DH	14.8	13.3	12.5	33.3	30.0
DM	-	-	-	-	-
EP	33.6	76.8	74.6	83.0	85.7
HD	91.2	94.1	94.3	95.0	95.6
JU	73.4	93.2	92.7	90.8	95.3
MNR	45.0	52.1	56.1	59.9	62.4
MO	65.3	71.1	76.0	77.5	78.6
NG	77.0	92.8	93.7	96.1	96.7
NK	92.5	93.4	95.1	95.9	96.2
NMC	78.4	93.6	95.8	96.7	100.2
OA	48.5	55.4	64.2	66.8	66.1
OA2	-	-	-	-	-
OC	53.9	54.5	57.7	60.4	60.5
OG	0.0	0.0	0.0	18.2	18.2
OP	15.1	11.7	25.9	55.2	52.4
PAR	39.3	35.8	39.0	48.1	48.0
PD	37.1	41.7	46.9	58.7	58.9
PG	15.8	10.2	49.3	63.3	63.3
PH	37.8	67.8	62.7	65.8	73.6
PM	95.9	97.5	97.7	97.5	98.0
PNC	72.2	77.9	81.5	82.8	84.0
RC	60.4	63.3	77.1	59.1	58.4
RE	29.1	34.4	23.9	33.7	33.9
RS	7.4	13.3	13.3	10.5	31.6
SB	68.8	73.3	79.6	78.6	79.4
SBP	14.3	8.5	57.1	77.2	74.2
SP	-	-	-	-	-
SVP	88.2	95.6	92.4	94.8	94.8
UC	0.0	44.4	43.1	40.7	54.2
TOTAL:	78.4	81.7	84.6	86.2	86.8

Table 10.5: F-scores for TiGer grammatical functions assigned by the different parsers and by the function labeller (TiGer DB)

cal function tags for the berk.fun and berk.fun.par settings are better than for the three parsers when trained on a combination of syntactic nodes and grammatical functions, but F-scores for F-structure evaluation for the two-step architecture of berk.fun and berk.par are substantially lower than for the combined approach. While precision for the two-step approach is around 2% higher than for the parser-assigned GFs, recall decreases dramatically to 62-64.7% for the development set and to 55-58% for the test set (Figure 10.1) for training on 25,000 trees (and similarly for training on 48,000 trees).

Below I take a look at the FunTag output for the gold standard-trained and the parser output-trained function labeller and discuss the differences in F-structures arising from the different input.

Looking at the most crucial differences in grammatical function labelling between the parser-assigned grammatical functions and the ones assigned by FunTag, we cannot find an explanation for the lower recall for F-structures in the two-step architecture. Table 10.8 shows results (accuracy: number of correctly labelled GFs / number of GFs in the gold standard) for grammatical functions occurring at least 100 times in the gold standard. For most of them (22 out of 28), FunTag F-scores are higher than results for the parser-assigned labels. Exceptions are genitive attributes (AG), comparative complements (CC), dative objects (DA), clausal objects (OC), relative clauses (RC) and subjects (SB).

The low recall in the FunTag output is not caused by incorrect function labelling, but by missing grammatical functions, violating the LFG completeness condition. Note that the evaluation in Tables 10.5 and 10.8 reports F-score and accuracy for those syntactic nodes only which have a corresponding node in the parser output. Evaluating grammatical functions is not straightforward. Following previous research in function labelling (Blaheta and Charniak, 2000; Chrupala et al., 2007), in order to know against what to evaluate, for each grammatical function label in the parser output which is attached to a syntactic node, we have to find a corresponding node in the gold tree. Table 10.6 shows the number of matching node instances found in both the gold standard and the parser output (matching nodes), the number of instances with a GF assigned by the parser or by FunTag (GF labels), and the number of correctly assigned GF labels (matching node-GF label pairs). There are almost 1000 more node-GF label pairs in the berk.fun and berk.fun.par settings having a corresponding node-GF label in the

setting	matching nodes	GF labels	matching node-GF label pairs
berk.par	38885	39256	36284
berk.fun	39889	39357	37103
berk.fun.par	39867	39039	37189

Table 10.6: GF evaluation: number of matching nodes in the gold standard and in the parser output (matching categorial nodes), number of GFs assigned in the test set (GF labels), number of correcly assigned GFs (matching node-GF-label pairs)

	gold	berk.par	berk.fun	berk.fun.par
all S nodes	2980	3001	2979	2979
S with GF	1399	1396	1067	998

Table 10.7: Number of S nodes with and without a GF in the gold trees and in the parser output

gold standard.

However, if we look at particular syntactic categories such as S, we find approximately the same number (about 3000) of S nodes in the gold standard and in the various parser outputs (Table 10.7). Out of these, 1399 S nodes in the gold standard are associated with a grammatical function (S nodes which are attached to the root node do not bear a grammatical function label). In the output of berk.par, where the function labelling was done by the parser, we have nearly the same number of grammatical function labels (1396), while for the two-step architecture (berk.fun and berk.fun.par) only about 1000 S nodes are assigned a grammatical function. S nodes without a GF label often fail to obtain the correct or in fact any LFG F-structure equation and are therefore often not included in the F-structure (and with them all child nodes of the S node), which drastically reduces recall for the two-step architecture (Berkeley/FunTag).

GF	**berk.par**		**berk.fun**		**berk.fun.par**	
AC	(3651/3688)	0.990	(3652/3688)	0.990	(3660/3688)	0.992
AG	(798/1049)	0.761	(773/1049)	0.737	(786/1049)	0.749
APP	(89/175)	0.509	(104/175)	0.594	(104/175)	0.594
CC	(48/103)	0.466	(33/103)	0.320	(32/103)	0.311
CD	(809/839)	0.964	(819/839)	0.976	(823/839)	0.981
CJ	(1392/2280)	0.611	(1584/2280)	0.695	(1578/2280)	0.692
CM	(72/104)	0.692	(74/104)	0.712	(74/104)	0.712
CP	(347/361)	0.961	(352/361)	0.975	(351/361)	0.972
DA	(80/195)	0.410	(75/195)	0.385	(74/195)	0.379
HD	(4912/5207)	0.943	(4933/5207)	0.947	(4966/5207)	0.954
MNR	(605/1075)	0.563	(614/1075)	0.571	(638/1075)	0.593
MO	(3478/4562)	0.762	(3503/4562)	0.768	(3485/4562)	0.764
NG	(230/244)	0.943	(232/244)	0.951	(237/244)	0.971
NK	(14860/15495)	0.959	(14869/15495)	0.960	(14918/15495)	0.963
NMC	(250/263)	0.951	(263/263)	1.000	(263/263)	1.000
OA	(880/1360)	0.647	(883/1360)	0.649	(855/1360)	0.629
OC	(884/1575)	0.561	(873/1575)	0.554	(874/1575)	0.555
OP	(72/343)	0.210	(150/343)	0.437	(136/343)	0.397
PAR	(40/138)	0.290	(51/138)	0.370	(49/138)	0.355
PD	(186/416)	0.447	(236/416)	0.567	(231/416)	0.555
PG	(56/115)	0.487	(75/115)	0.652	(76/115)	0.661
PH	(74/131)	0.565	(74/131)	0.565	(89/131)	0.679
PM	(195/203)	0.961	(196/203)	0.966	(197/203)	0.970
PNC	(848/1045)	0.811	(870/1045)	0.833	(868/1045)	0.831
RC	(200/276)	0.725	(137/276)	0.496	(131/276)	0.475
RE	(23/122)	0.189	(25/122)	0.205	(25/122)	0.205
SB	(2083/2661)	0.783	(2047/2661)	0.769	(2046/2661)	0.769
SVP	(194/208)	0.933	(199/208)	0.957	(199/208)	0.957
TOTAL	(37562/44681)	0.841	(37931/44681)	0.849	(38013/44681)	0.851

Table 10.8: Accuracy for grammatical functions assigned by the Berkeley parser (berk.par) and in the two-step architecture (berk.fun, berk.fun.par) (TiGer DB)

	bitpar	stanford	berk.par	berk.fun	berk.fun.par
TüBa-D/Z-25000 - c-structure evaluation					
# sent < 40	98	98	98	98	98
# parse	98	98	98	98	98
F-score no GF	84.4	86.6	89.3	89.2	89.2
F-score GF	72.7	75.5	80.2	76.3	76.0
tagging acc.	94.7	96.4	96.5	96.4	96.4
TüBa-D/Z F-structure evaluation					
# sent	100	100	100	100	100
% f-struct.	98.0	96.0	96.0	99.0	99.0
Precision	68.2	73.6	76.9	75.8	77.0
Recall	42.0	41.1	45.1	39.3	34.5
F-score	52.0	52.7	56.9	51.7	47.7
TiGer25000 - F-structure evaluation					
# sent	100	100	100	100	100
% f-struct.	93.0	95.0	94.0	98.0	94.0
Precision	66.5	70.0	72.9	76.4	77.8
Recall	66.3	67.5	70.9	61.3	60.8
F-score	66.4	68.7	71.8	68.0	68.2
TiGer48000 - F-structure evaluation					
# sent	100	100	100	100	100
% f-struct.	93.0	96.0	89.0	95.0	90.0
Precision	68.7	72.1	73.3	76.1	75.9
Recall	69.8	71.4	70.6	58.7	59.9
F-score	69.2	71.7	72.0	66.3	64.4

Table 10.9: TüBa-D/Z c-structure and TüBa-D/Z / TiGer F-structure evaluation for different German grammars and parser (TUBA100)

10.3.4 C-Structure and F-Structure Parsing Results for the TüBa-D/Z

In Chapter 6 I investigated the impact of treebank design on PCFG parsing. In this section I present a task-based evaluation of the treebanks by comparing the suitability of TiGer and TüBa-D/Z for the automatic acquisition of LFG resources.

Table 10.9 presents parsing results for c-structures and F-structures for the TüBa-D/Z and TiGer trained parsers (with and without FunTag) against TUBA-100. `Evalb` results for the TüBa-D/Z-trained parser outputs are, as usual, far higher than the ones for TiGer, with F-scores in the range of 84.4% (bitpar, noGF) to 89.3% (berk.par, noGF). Training on TüBa-D/Z, the Berkeley parser yields slightly higher results when trained on syntactic nodes including grammatical

functions (berk.par, noGF: 89.3% vs. berk.fun, noGF: 89.2%), but considering the small size of the TUBA100 test set we should take this with a grain of salt.

At the level of F-structure we can now compare results for F-structures generated from the output of the three parsers trained on TiGer (25,000 and 48,000 trees) and on the TüBa-D/Z. The TiGer-trained parser output has been annotated with a version of the annotation algorithm adapted to the TiGer DB, for the TüBa-D/Z-trained parser output I used the TUBA100-style annotation algorithm. Looking at precision, results for the two versions of the annotation algorithm are quite similar. For F-structures annotated with the TUBA100-style annotation algorithm on TüBa-D/Z-trained parser output, however, recall is dramatically low. This is partly due to the small size of the TUBA100, which is not sufficient as a development/test set for grammar development. However, there are other reasons, too.

For the FunTag approach, the same problem we encountered when assigning TiGer treebank-style grammatical functions applies to the TüBa-D/Z, too. Due to missing grammatical function labels in the FunTag output, recall for the two-step architecture is much lower than for the setting where GF tags are assigned by the parser. Furthermore, we also observe a very low recall for F-structures generated from parser output from the TüBa-D/Z-trained parsers (bitpar, stanford, berk.par). In addition to the restricted size of the TUBA100, there are problems with regard to the annotation scheme of the TüBa-D/Z for treebank-based grammar acquisition.

One problem is caused by the TüBa-D/Z annotation scheme, where phrases which do not display a clear dependency relation to the other constituents in the tree are simply attached directly to the virtual root node. Arguably this treatment is suitable for phrases separated by a colon or a dash (Figure 10.9), but is widely applied to other phrases, too (Figure 10.10). In contrast to this, the TiGer annotation scheme would annotate the adjectival phrase *exzellent gespielt von Catherine Deneuve* (brilliantly performed by Catherine Deneuve) in Figure 10.10 as a sister node of the NP and assign the label APP (apposition). The TüBa-D/Z annotation scheme results in crossing branches (which have to be resolved; see Figure 10.10), and the final tree structure makes it impossible for the LFG F-structure annotation algorithm do disambiguate the sentence and find a suitable dependency relation for the node attached to the root node. In

(28) Landesvorsitzende Ute Wedemeier : Ein Buchungsfehler
state executive president Ute Wedemeier : an accounting error

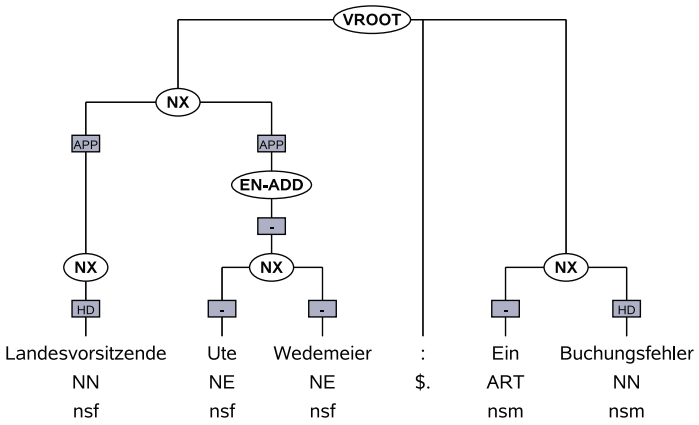


Figure 10.9: High attachment for independent phrases in TüBa-D/Z

most cases this TüBa-D/Z annotation practice cannot be resolved and so phrases attached high are often not represented in the F-structure, and this contributes to the low recall for the TüBa-D/Z F-structures.

Another problem is caused by the high degree of underspecification in the TüBa-D/Z annotation. The label MOD, for example, describes an ambiguous modifier. It is not possible to determine which node is modified by a MOD-labelled node. The MOD label occurs with high frequency in the TüBa-D/Z (> 24,300).

(30) (NX (NX-HD 150 000 Mark) (NX– Sammelgelder))
150 000 mark charity money$_{NOM}$

(31) (NX (NX-HD der Vorstand) (NX– der Wohlfahrtsorganisation))
the management (of) the$_{GEN}$ charity organisation$_{GEN}$

(32) (NX (NX-HD Friede) (NX– den Hütten))
peace (for) the$_{DAT}$ barracks$_{DAT}$

(33) (NX (NX-HD ein Dogmatiker) (NX– wie Perot))
a dogmatist like Perot$_{NOM}$

Another case of underspecification is the annotation of appositions in the TüBa-D/Z (see Section 5.3.1). The same is true for TüBa-D/Z internal NP structure

(29) Ein Krimistück mit feinem , melancholischem Ton , in dem eine
A murder mystery with fine , melancholic tone , in which a
Frau , exzellent gespielt von Catherine Deneuve , wieder zu Sinnen kommt
woman , excellent played by Catherine Deneuve , again to senses comes
A murder mystery with a subtle, melancholic note, in which a woman, brilliantly performed by Catherine Deneuve, comes to her right mind

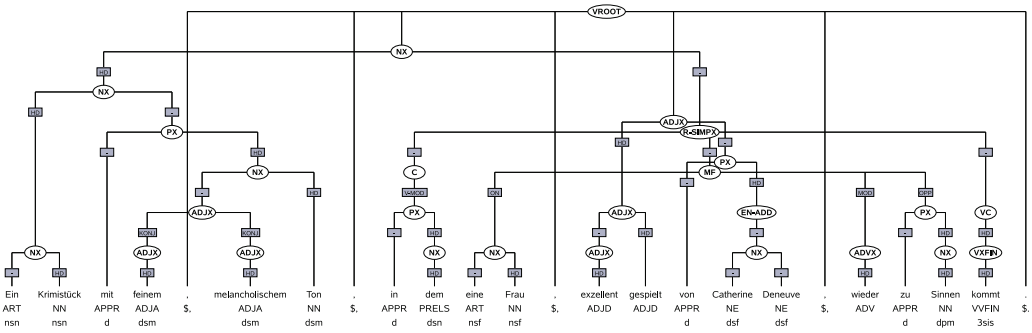


Figure 10.10: High attachment for independent phrases in TüBa-D/Z

in general. At first glance it seems as if the TüBa-D/Z annotation of NPs and PPs is more specific than the one in TiGer, because TüBa-D/Z explicitly marks the head (HD) of an NP, or the prepositional object NP inside a PP, while TiGer uses the underspecified label NK (noun kernel) for all nouns, adjectives and determiners attached to the NP or PP. However, examples (30-33) (TüBa-D/Z) and (34-37) (TiGer) show that, despite the head annotation in the TüBa-D/Z, the TüBa-D/Z trees reveal less information than the TiGer trees. In the TiGer annotation scheme, the second constituent in (31) e.g. would be annotated with the grammatical function label AG (genitive attribute, as in (35)), (32) would obtain the label DA (dative attribute, as in (36)), and (33) would be annotated as comparative complement as in (37). In the TüBa-D/Z, all four examples (30-33) exhibit the same tree structure, with the second NP (NX) assigned the default label '-' (non-head). Part of the missing information can be retrieved from morphological annotations, but this would require an extensive treebank transformation and probably result in a sparse data problem. For parser output trees morphological information is not in general available. Moreover, the focus of this thesis is on investigating treebank design and its impact on parsing and

LFG grammar acquisition. As things stand, the grammars extracted from TiGer encode more specific information than the TüBa-D/Z grammars.

(34) (NP (NM-NK 150 000) (NN-NK Mark) (NN-NK Sammelgelder))
150 000 mark charity money$_{NOM}$

(35) (NP (ART-NK der) (NN-NK Vorstand) (NP-AG der
the management (of) the$_{GEN}$ charity organisation$_{GEN}$
Wohlfahrtsorganisation))

(36) (NP (NN-NK Friede) (NP-DA den Hütten))
peace (for) the$_{DAT}$ barracks$_{DAT}$

(37) (NP (ART-NK ein) (NN-NK Dogmatiker) (NP-CC wie Perot))
a dogmatist like Perot$_{NOM}$

Moving on to the annotation of grammatical functions in the trees, we note a substantial gap between TiGer and TüBa-D/Z (Table 10.10). In the original treebanks we have roughly the same number of grammatical functions (44 in TiGer versus 40 in the TüBa-D/Z). In the parser/FunTag output of the TiGer-trained parsers on the TiGer CCG2000 test set (fully presented in Section 10.3.5 below) there are between 39 and 41 different grammatical functions, a number only slightly lower than the one attested in the TiGer CCG2000 gold test set (42), while the number of different TüBa-D/Z GF labels in TüBa-D/Z-trained parser output against the CCG2000 gold standard is far smaller with 33 different grammatical function labels. The number of different GF labels reproduced by the different TüBa-D/Z-trained parsers and FunTag varies widely. While the Berkeley parser trained on a combination of syntactic nodes and grammatical functions assigns 31 different GFs (a number close to the one in the gold test set), the other parsers perform considerably worse with 27 (BitPar) and 24 (Stanford) different grammatical function labels. The SVM-based function labeller performs even worse. In both settings (berk.fun, berk.par) FunTag only manages to reproduce 19 of the original TüBa-D/Z GF labels.

This raises the suspicion that the design of the GF label set in the TüBa-D/Z is not optimal for machine learning-based NLP applications, such as data-driven parsing, function labelling and grammar acquisition. As already mentioned in Section 2.2, many of the grammatical functions in the TüBa-D/Z occur with a low frequency only (e.g. OA-MODK, ON-MODK, OADVPK, OG-MOD,

CCG2000	gold	bitpar	stanford	berk.par	berk.fun	berk.fun.par
TiGer	42	41	42	41	40	39
TüBa-D/Z	33	27	24	31	19	19

Table 10.10: Number of different grammatical functions in the TiGer/TüBa-D/Z CCG2000 test set and reproduced by the different parsers and FunTag

OADJP-MO, OADVP-MO, FOPPK, MODK), which makes it extremely difficult for statistical methods to learn these labels.

Table 10.11 shows F-scores for TüBa-D/Z grammatical function labelling for the TüBa CCG2000 test set. With the exception of BitPar, which shows better results on the TüBa-D/Z test set (compare Table 10.5 for TiGer GF results), all other parsers perform better on assigning TiGer grammatical functions. Comparing performance for the main grammatical functions (subject (ON), accusative object (OA) and dative object (DA), Table 10.12), Table 10.11 shows that for dative objects the TüBa-D/Z-trained Berkeley parser outperforms the TiGer-trained parsing model (one-step architecture), while for all other grammatical functions we obtain better results for TiGer. Again the SVM-based FunTag shows poor performance on the TüBa-D/Z data, while for TiGer the function labeller trained on parser output (berk.fun.par) outperforms all other GF labelling approaches on subjects, accusative and dative objects.

The asymmetric behaviour of FunTag (TiGer vs. TüBa-D/Z) might be due to the different data structures in the treebanks. It seems as if the topological fields in TüBa-D/Z remove necessary contextual information, which would otherwise be encoded in the FunTag training feature set.

10.3.5 C-Structure and F-Structure Parsing Results in a CCG-Style Evaluation

In order to put the (potentially preliminary) results on the small (hand-crafted) TüBa-D/Z test set TUBA100 into perspective, I complement the evaluation with a CCG-style experiment Hockenmaier (2003), where I evaluate on a larger test set of 2000 sentences (TiGer CCG2000 and TüBa CCG2000) from both TiGer and TüBa-D/Z. The CCG-style gold standard is generated automatically by applying

GF	bitpar		stanford		berk.par		berk.fun		berk.fun.par	
APP	(111/543)	0.279	(314/674)	0.545	(557/708)	0.818	(167/234)	0.708	(129/228)	0.640
ES	(0/6)	-	(0/6)	-	(0/6)	-	(0/4)	-	(0/4)	-
FOPP	(21/241)	0.130	(24/276)	0.136	(96/282)	0.374	(20/86)	0.323	(15/86)	0.265
FOPPK	(0/0)	-	(0/1)	-	(0/1)	-	(0/0)	-	(0/0)	-
FOPPMOD	(0/9)	-	(0/5)	-	(0/9)	-	(0/5)	-	(0/5)	-
HD	(24707/25429)	0.970	(25790/26293)	0.974	(26181/26600)	0.984	(9532/9874)	0.965	(9534/9858)	0.967
KONJ	(1269/1512)	0.840	(1329/1583)	0.852	(1552/1759)	0.884	(434/570)	0.783	(422/559)	0.789
MOD	(1201/1517)	0.703	(1218/1628)	0.716	(1388/1683)	0.801	(423/601)	0.752	(418/601)	0.757
MODMOD	(4/11)	0.400	(4/16)	0.235	(6/17)	0.387	(0/7)	-	(0/7)	-
OA	(613/1109)	0.540	(669/1193)	0.593	(879/1223)	0.720	(189/442)	0.482	(170/442)	0.466
OADJP	(0/8)	-	(0/6)	-	(0/10)	-	(0/3)	-	(0/3)	-
OADVP	(0/9)	-	(0/11)	-	(1/10)	0.154	(1/4)	0.400	(1/4)	0.400
OADVPMO	(0/0)	-	(0/1)	-	(0/1)	-	(0/0)	-	(0/0)	-
OAK	(0/1)	-	(0/1)	-	(0/1)	-	(0/0)	-	(0/0)	-
OAMOD	(0/42)	-	(0/76)	-	(8/94)	0.119	(0/30)	-	(0/30)	-
OD	(22/161)	0.190	(25/169)	0.215	(84/171)	0.575	(11/62)	0.275	(11/62)	0.275
ODMOD	(0/0)	-	(0/2)	-	(0/2)	-	(0/1)	-	(0/1)	-
OG	(0/4)	-	(0/4)	-	(0/3)	-	(0/1)	-	(0/1)	-
ON	(2000/2350)	0.776	(1992/2445)	0.782	(2244/2525)	0.861	(758/1000)	0.723	(725/1000)	0.728
ONK	(0/1)	-	(0/1)	-	(0/2)	-	(0/1)	-	(0/1)	-
ONMOD	(2/71)	0.048	(6/86)	0.116	(10/97)	0.145	(0/39)	-	(0/39)	-
OPP	(96/274)	0.344	(98/317)	0.326	(153/326)	0.453	(24/106)	0.329	(16/106)	0.254
OPPK	(0/1)	-	(0/1)	-	(0/1)	-	(0/0)	-	(0/0)	-
OPPMOD	(0/16)	-	(0/14)	-	(0/20)	-	(0/5)	-	(0/5)	-
OS	(79/166)	0.532	(126/195)	0.604	(165/208)	0.637	(21/62)	0.359	(21/62)	0.385
OSMOD	(0/8)	-	(0/8)	-	(1/8)	0.182	(0/3)	-	(0/3)	-
OV	(945/980)	0.956	(930/1004)	0.941	(973/991)	0.975	(310/332)	0.944	(313/332)	0.946
PRED	(149/398)	0.456	(170/455)	0.462	(242/457)	0.565	(53/183)	0.406	(51/183)	0.394
PREDK	(0/1)	-	(1/1)	1.000	(1/2)	0.667	(0/0)	-	(0/0)	-
PREDMOD	(1/7)	0.154	(0/16)	-	(0/22)	-	(0/12)	-	(0/12)	-
VMOD	(682/1169)	0.524	(750/1266)	0.554	(955/1361)	0.680	(299/450)	0.657	(280/450)	0.662
VMODK	(1/1)	1.000	(0/0)	-	(0/1)	-	(0/1)	-	(0/1)	-
VPT	(179/179)	1.000	(180/180)	1.000	(181/181)	1.000	(56/56)	1.000	(56/56)	1.000
Total		0.881		0.889		0.919		0.883		0.886

Table 10.11: F-scores for grammatical functions assigned by the different parsers and by the function labeller (TüBa-D/Z, CCG2000)

GF	bitpar	stan	berk.par	berk.fun	berk.fun.par
TiGer25000 - GF evaluation					
DA	20.0	31.3	52.5	75.9	77.1
OA	67.5	70.9	79.5	85.3	87.0
SB	82.9	84.3	90.0	88.7	91.9
All GF	90.0	90.9	93.1	94.5	95.6
TüBa-D/Z-25000 - GF evaluation					
OD	19.0	21.3	56.8	46.1	45.4
OA	52.8	57.1	69.0	58.1	56.0
ON	77.4	77.9	85.2	80.8	81.3
All GF	88.1	88.9	91.9	87.1	87.5

Table 10.12: Evaluation of main grammatical functions in TiGer and TüBa-D/Z (dative object: DA/OD, accusative object: OA, prepositional object: OP, subject: SB/ON) on the CCG2000 test set

the LFG F-structure annotation algorithm to gold treebank trees. I evaluate the *parser output* F-structures against the automatically generated *gold tree* F-structures. The CCG-style evaluation provides a fairer basis for comparing the results for the different versions of the annotation algorithm. I expect that the larger size of the TiGer DB gold standard (both development and test sets) helped to improve results for TiGer treebank-based F-structure annotation, especially for recall. The CCG-style experiment should, at least partly, make up for this, as the F-structures are evaluated against automatically annotated F-structures from gold tree input. This means that grammar phenomena which did not occur in the gold standard (development sets) and thus cannot be dealt with by the annotation algorithm are excluded from the evaluation.

Table 10.13 shows `evalb` results for c-structures and F-structures for TiGer and TüBa-D/Z. We observe the same parser ranking as before (BitPar > Stanford > Berkeley), and again the Berkeley parser gives the best constituency results for the TiGer training set when trained on syntactic nodes only (berk.fun, berk.fun.par), while for the TüBa-D/Z data the parser trained on a combination of syntactic node labels with grammatical functions gives slightly better results (berk.par). This confirms our findings from the TUBA100-based TüBa-D/Z eval-

	bitpar	stanford	berk.par	berk.fun	berk.fun.par
TiGer25000 - c-structure evaluation					
# sent <= 40	1939	1939	1939	1939	1939
# parses	1935	1938	1935	1937	1937
F-score noGF	73.9	75.7	80.6	82.4	82.4
F-score GF	62.7	64.2	71.0	73.5	74.3
tagging acc.	95.8	97.3	96.3	96.8	96.8
TiGer25000 - F-structure evaluation (CCG-style)					
# sent	2000	2000	2000	2000	2000
% f-struct.	91.3	92.0	92.0	95.3	93.4
Precision	79.2	81.9	84.5	87.9	88.6
Recall	79.2	80.7	84.0	72.6	69.8
F-score	79.2	81.3	84.2	79.5	78.1
TüBa-D/Z-25000 - c-structure evaluation					
# sent <= 40	1929	1929	1929	1929	1929
# parses	1927	1927	1911	1927	1927
F-score	87.2	88.3	91.5	90.9	90.9
F-score GF	73.4	77.1	83.2	78.1	77.6
tagging acc.	94.6	96.4	96.7	96.6	96.6
TüBa-D/Z-25000 - F-structure evaluation (CCG-style)					
# sent <= 40	2000	2000	2000	2000	2000
% f-struct.	90.5	91.3	92.4	92.1	90.6
Precision	73.6	77.3	81.0	81.1	81.7
Recall	45.3	46.1	52.0	38.7	35.4
F-score	56.1	57.7	63.3	52.4	49.4

Table 10.13: C-structure parsing results (labelled F-score) and F-structure evaluation for different TiGer and TüBa-D/Z grammars and parser (CCG-style)

uation in the last section.

On the F-structure level, F-scores for the CCG-style evaluation are clearly higher than for evaluating against the different hand-crafted gold standards (Table 10.14). This is not so much due to a higher precision (in fact results for the DCU250 gold standard for the TiGer-trained parsers, to take but one example, are only around 1-3% lower), but to a better recall, resulting from the fact that some constructions causing a clash when evaluating against the F-structures for the hand-crafted gold standards are missing in the automatically generated CCG-style gold standard, too. F-structures generated from the output of TüBa-D/Z-trained parsers show lower precision than for TiGer-trained parsers, but even here best results are still over 80%. Recall, however, is again very low with a best score of 52% for the TüBa-D/Z-trained Berkeley parser (berk.par), most likely due to the limited size of the TUBA100 development set for constructing

GF	TiGer (berk.par)			TüBa-D/Z (berk.par)		
	prec.	rec.	f-sc.	prec.	rec.	f-sc.
adj_gen	823/931=88	823/963=85	87	232/269=86	232/636=36	51
adj_rel	107/246=43	107/236=45	44	47/187=25	47/144=33	28
ams	11/14=79	11/23=48	59	0/0=0	0/3=0	0
app	301/433=70	301/436=69	69	99/134=74	99/430=23	35
app_clause	12/82=15	12/97=12	13			
circ_form	6/11=55	6/7=86	67	0/0=0	0/1=0	0
comp	127/244=52	127/205=62	57	502/635=79	502/668=75	77
comp_form	94/119=79	94/116=81	80	51/111=46	51/68=75	57
conj	1342/1727=78	1342/1807=74	76	882/1188=74	882/1697=52	61
coord_form	657/694=95	657/717=92	93	414/455=91	414/651=64	75
da	64/159=40	64/156=41	41	58/109=53	58/150=39	45
det	3941/4054=97	3941/4065=97	97	2056/2160=95	2056/3628=57	71
det_type	3979/4008=99	3979/4026=99	99	2135/2181=98	2135/3679=58	73
fut	4/5=80	4/6=67	73	46/51=90	46/60=77	83
measured	5/5=100	5/7=71	83	1/2=50	1/8=12	20
mo	5056/7048=72	5056/7036=72	72	2256/2847=79	2256/6387=35	49
mo_type	177/179=99	177/181=98	98			
mod	37/44=84	37/40=92	88	12/15=80	12/27=44	57
name_mod	417/467=89	417/480=87	88	137/147=93	137/516=27	41
number	293/360=81	293/362=81	81	108/150=72	108/259=42	53
oa	827/1196=69	827/1175=70	70	656/1114=59	656/1127=58	59
obj	3340/3527=95	3340/3531=95	95	837/923=91	837/3076=27	42
obj_gen	1/5=20	1/11=9	13			
obl_compar	14/39=36	14/58=24	29			
op	85/233=36	85/317=27	31	107/284=38	107/311=34	36
part_form	172/192=90	172/187=92	91			
pass_asp	99/104=95	99/103=96	96	196/219=89	196/225=87	88
pd	177/296=60	177/328=54	57	193/364=53	193/449=43	47
perf	34/38=89	34/36=94	92	208/219=95	208/239=87	91
poss	268/281=95	268/282=95	95	161/174=93	161/249=65	76
postcoord_form	8/22=36	8/12=67	47	6/11=55	6/17=35	43
precoord_form	7/8=88	7/7=100	93	4/4=100	4/7=57	73
pred_restr	6/17=35	6/9=67	46			
pron_form	43/49=88	43/45=96	91	87/94=93	87/98=89	91
pron_type	1078/1212=89	1078/1236=87	88	1401/1492=94	1401/1685=83	88
quant	278/310=90	278/319=87	88	49/57=86	49/227=22	35
sb	3239/3870=84	3239/3946=82	83	2050/2704=76	2050/3178=65	70
sbp	34/49=69	34/51=67	68			
tiger_id	1672/1778=94	1672/1812=92	93	1636/2020=81	1636/2231=73	77
xcomp	909/1114=82	909/1045=87	84	102/199=51	102/159=64	57

Table 10.14: Dependency relations for TiGer and TüBa-D/Z (CCG-style, berk.par)

the TüBa-D/Z annotation algorithm and the TüBa-D/Z representation and annotation design problems identified in Section 10.3.4. The CCG-style experiment confirms the results from the evaluation on the small TUBA100 test set on a much larger data set. The overall best result is an F-structure F-score of 84.2% for the TiGer-trained Berkeley parser (setting berk.par).

10.3.6 LFG F-structure Annotation with TiGer and TüBa-D/Z Trained Parsing Resources - Conclusions

So far the results of our experiments indicate that the annotation scheme of the TiGer treebank is more adequate for the automatic acquisition of LFG resources and treebank-based parsing into LFG representations. The GF label set in the TüBa-D/Z has been designed with the secondary aim of expressing non-local dependencies between nodes, while the TiGer grammatical functions focus solely on encoding more detailed linguistic information about the grammatical function of the node itself. Therefore one might assume that, despite encoding less fine-grained linguistic information, the TüBa-D/Z approach to encode non-local dependencies with the help of grammatical function labels is superior to the treatment in TiGer, where the same information is expressed through crossing branches, which have to be resolved before parsing and so can result in a loss of information. However, this is only true if the TüBa-D/Z grammatical functions expressing non-local dependencies can be reproduced by a parser or a function labeller with sufficient reliability and coverage. If this is not possible, the TüBa-D/Z way of annotating grammatical functions seems less suitable than the one in TiGer.

Other potential problems for LFG F-structure annotation on TüBa-D/Z trees have already been addressed in Chapter 6. The parser-based F-structure evaluations presented in this chapter give further evidence for the difficulties arising from the more hierarchical (and hence in a sense less transparent) structure of the TüBa-D/Z. To give just one example: in the TüBa-D/Z-style F-structures for the different parsers/settings, none of the 9 relative clauses (rc) in the TUBA100 (Table 10.15) were identified, while for the TiGer-style F-structures between 2 and 4 of the 11 relative clauses in the TUBA100 were annotated correctly in the F-structures.

Overall, it seems as if treebank-based grammar acquisition for the TüBa-D/Z in general is possible, but raises serious problems. The annotation scheme of the TüBa-D/Z seems to be less adequate to support our approach of LFG-based grammar acquisition and parsing, and a number of important problems have to be addressed, especially for increasing recall, before we can expect high-quality results for treebank-based acquisition of LFG resources based on the TüBa-D/Z treebank.

10.4 Summary

This chapter presents an extensive evaluation of the different grammar acquisition and parsing architectures, using different parsers and FunTag, an automatic grammatical function labeller. I compared performance for the system based on two different German treebanks. Results for the different gold standards and training sets show the same general trends:

- All experiments result in the same parser ranking: BitPar < Stanford < Berkeley.
- For constituent-based evaluation (`evalb`), the TiGer treebank-trained Berkeley parser trained on syntactic nodes only outperforms the same parser trained on a combination of syntactic nodes and grammatical function labels, while TüBa-D/Z-trained parsers achieve better results when trained on a combination of syntactic categories and grammatical function labels.
- For a parser trained on TiGer syntactic nodes without grammatical functions, enlarging the size of the training data does not improve parsing performance significantly. For a parser trained on TiGer syntactic nodes merged with grammatical functions, increased training sets may produce improved results.
- While precision for F-structures generated from Berkeley parser output is quite high, recall is still a major problem, especially for the two-step architecture (Berkeley/FunTag), but also for Tüba-D/Z-generated F-structures.

TüBa-D/Z-25000										
GF	**bitpar**		**stanford**		**berk.par**		**berk.fun**		**berk.fun.par**	
	(prec/rec)	**F-score**	**(prec/rec)**	**F-score**	**(prec/rec)**	**F-score**	**(prec/rec)**	**F-score**	**(prec/rec)**	**F-score**
ams	(100/100)	100	(0/0)	0	(0/0)	0	(0/0)	0	(0/0)	0
app	(20/4)	7	(50/9)	15	(40/18)	25	(38/14)	20	(50/9)	15
app-cl	(0/0)	0	(0/0)	0	(0/0)	0	(0/0)	0	(0/0)	0
cc	(100/20)	33	(100/40)	57	(100/20)	33	(0/0)	0	(0/0)	0
cj	(69/32)	43	(69/26)	38	(67/43)	52	(72/30)	42	(71/29)	41
comp-form	(67/40)	50	(100/20)	33	(100/80)	89	(100/20)	33	(100/20)	33
coord-form	(86/46)	60	(94/37)	53	(83/50)	62	(92/29)	44	(92/29)	44
da	(0/0)	0	(0/0)	0	(25/25)	25	(0/0)	0	(0/0)	0
degree	(0/0)	0	(0/0)	0	(0/0)	0	(0/0)	0	(0/0)	0
det	(100/53)	70	(100/50)	67	(99/59)	74	(100/40)	57	(100/28)	44
det-type	(98/54)	70	(98/51)	67	(99/61)	75	(97/41)	57	(96/28)	44
fut	(100/67)	80	(100/50)	67	(100/67)	80	(67/67)	67	(67/67)	67
gl	(0/0)	0	(0/0)	0	(0/0)	0	(0/0)	0	(0/0)	0
gr	(100/32)	48	(82/37)	51	(86/51)	64	(88/37)	52	(100/24)	38
measured	(0/0)	0	(0/0)	0	(100/100)	100	(0/0)	0	(0/0)	0
mo	(64/27)	38	(77/27)	40	(71/32)	44	(68/23)	34	(66/19)	30
mod	(100/25)	40	(100/50)	67	(0/0)	0	(0/0)	0	(0/0)	0
name-mod	(100/7)	12	(100/36)	53	(85/38)	52	(100/25)	40	(100/14)	25
number	(100/25)	40	(75/19)	30	(100/19)	32	(100/12)	22	(50/6)	11
oa	(37/48)	42	(50/52)	51	(66/71)	69	(60/60)	60	(69/52)	59
obj	(80/19)	31	(90/21)	34	(93/30)	45	(93/15)	25	(91/11)	20
oc-fin	(62/45)	53	(29/36)	32	(42/45)	43	(75/55)	63	(67/55)	60
oc-inf	(50/75)	60	(62/75)	68	(61/64)	62	(77/71)	74	(81/71)	76
op	(28/28)	28	(16/17)	16	(38/44)	41	(33/11)	17	(67/11)	19
pass-asp	(56/69)	62	(67/77)	71	(71/77)	74	(89/62)	73	(89/62)	73
pd	(38/21)	27	(33/14)	20	(38/29)	33	(44/24)	31	(62/27)	37
perf	(100/64)	78	(100/64)	78	(100/73)	84	(100/55)	71	(100/55)	71
pron-form	(100/100)	100	(100/100)	100	(100/100)	100	(100/100)	100	(100/100)	100
pron-type	(96/85)	90	(96/88)	92	(89/84)	86	(98/75)	85	(98/75)	85
quant	(60/17)	26	(60/17)	26	(75/33)	46	(75/17)	27	(75/17)	27
rc	(0/0)	0	(0/0)	0	(0/0)	0	(0/0)	0	(0/0)	0
sb	(69/55)	61	(80/54)	65	(77/62)	68	(84/54)	66	(87/52)	65
tiger-id	(76/77)	76	(74/74)	74	(79/79)	79	(80/83)	81	(80/79)	80
TOTAL	(68.2/42.0)	52.0	(73.6/41.1)	52.7	(72.9/49.1)	58.6	(75.6/37.6)	50.2	(76.7/32.9)	46.0

Table 10.15: F-scores for F-structure annotation on different parser output and by the function labeller (TUBA100)

Comparing results for the different treebanks, I show that TüBa-D/Z-based dependency results are significantly lower than the ones for the TiGer-based architecture. Even when evaluated against the TUBA100 gold standard, results for F-structures generated under the TiGer treebank-based architecture are higher than the ones achieved in the TüBa-D/Z-based architecture. To be sure, this is partly due to the limited size of the data set used for grammar development, but also an artifact of the annotation scheme of the TüBa-D/Z: one major drawback follows from the more hierarchical tree structure, which results in data structures which are less transparent for PCFG parsers, because relevant information is embedded deep in the tree and is not captured in the local context encoded in the grammar rules. Another problem is caused by the high degree of underspecification in the TüBa-D/Z. Nodes which, due to ambiguous dependencies, have been attached high up at the root of the tree do not contribute meaningful dependencies and add to the low recall scores for the TüBa-D/Z. Finally, the TüBa-D/Z design decision to encode non-local dependencies with the help of grammatical function labels is not optimal to support PCFG parsing. The parsers have considerable difficulties to learn these labels, which can be seen by the low overall number of different labels reproduced in the parser output, as well as by the modest results for grammatical function labelling for parser output and for the SVM-based grammatical function labelling software.

As a result of the problems for GF label learning, non-local dependencies are not represented adequately in the TüBa-D/Z parser output. In TiGer, the conversion to CFG trees by raising the non-head child nodes of discontiguous trees results in a loss of information. However, the flat annotation yields some transparency and allows us to recover at least some of the non-local dependencies, while for the TüBa-D/Z this is not possible.

In the next Chapter I present two extensions to the LFG grammar acquisition: the recovery of LDDs in the parse trees and a method for improving coverage, based on subcat frames automatically extracted from LFG F-structures.

Chapter 11

Extensions: Recovering LDDs and Improving Coverage with SubCat Frames

11.1 Introduction

Chapter 10 presented parsing experiments using the automatic F-structure annotation algorithm described in Chapter 9. Evaluation results showed good precision for the automatically generated F-structures. However, a number of problems have become apparent in the evaluation:

- low recall especially for F-structures automatically generated from TüBa-D/Z-trained parser output;
- low recall for F-structures automatically generated from the two-step architecture due to missing GFs in the FunTag output;
- low coverage (% of F-structures) due to clashes in the constraint solver, caused by conflicting grammatical functions assigned by the parser or FunTag;
- missing long distance dependencies (LDDs) due to the raising-based resolution (Kübler, 2005) of crossing branches in TiGer, resulting in shallow "proto" F-structures.

This chapter addresses two of these problems, namely the low coverage and missing long-distance dependencies in the F-structures derived from the raising approach to convert crossing branches into CFG trees to train parsers. First I apply Boyd (2007)'s split node method for converting discontiguous trees into CFG representations, and compare the performance of the raised node Kübler (2005) and split node Boyd (2007) conversion methods on F-structure level. Then I present a method to improve coverage using automatically extracted subcategorisation frames.

11.2 Recovering LDDs in the Parse Trees

Chapter 10 evaluated F-structures generated from the TiGer parser output where crossing branches were resolved using the raised-node conversion method. This results in shallow F-structures with long-distance dependencies unresolved. For the TüBa-D/Z, results for GF labelling are clearly not good enough to support meaningful resolution of LDDs based on the grammatical function labels in the parser output trees.

In this Section I will look at F-structures generated from parser output from a parser trained on a version of TiGer, where discontiguous trees have been resolved by inserting partial nodes in the trees (split-node conversion). Boyd (2007) performs a labeled dependency-based evaluation and reports a significant improvement for subjects, accusative objects, dative objects and prepositional objects for the improved representation of non-local dependencies in the tree.

I applied Boyd (2007)'s method to the large TiGer training set (48,000 sentences) and trained the Berkeley parser on the data, where syntactic nodes and grammatical functions were merged into new atomic labels. Tables 11.1 and 11.2 show results for F-structures generated from Berkeley parser output from raised-node (Kübler, 2005) and split-node converted versions of TiGer.

For both development and test set, results for the Berkeley parser without partial node annotation are slightly higher. For some dependencies, however, we observe a substantial improvement when using Boyd's technique. F-scores for the annotation of relative clauses, for example, rise from 36% to 45% for the development set, and from 33% to 46% for the test set. Results for dative objects are also better with 46% vs. 50% (development set) and 46% vs. 51%

GF	berk.raised			berk.split		
	prec.	rec.	f-sc.	prec.	rec.	f-sc.
ams	(5/7) 71	(5/8) 62	67	(5/9) 56	(5/7) 71	63
app	(174/403) 43	(174/271) 64	52	(180/384) 47	(180/263) 68	56
app_cl	(13/53) 25	(13/58) 22	23	(15/25) 60	(15/56) 27	37
cc	(4/37) 11	(4/28) 14	12	(6/17) 35	(6/31) 19	25
circ_form	(5/9) 56	(5/5) 100	71	(4/8) 50	(4/4) 100	67
cj	(955/1301) 73	(955/1363) 70	72	(854/1173) 73	(854/1206) 71	72
comp_form	(93/103) 90	(93/109) 85	88	(77/87) 89	(77/99) 78	83
coord_form	(470/502) 94	(470/516) 91	92	(419/447) 94	(419/456) 92	93
da	(44/94) 47	(44/96) 46	46	(45/89) 51	(45/90) 50	50
det	(2899/3100) 94	(2899/3132) 93	93	(2665/2842) 94	(2665/2904) 92	93
det_type	(2953/3080) 96	(2953/3026) 98	97	(2719/2832) 96	(2719/2808) 97	96
fut	(44/51) 86	(44/47) 94	90	(44/46) 96	(44/50) 88	92
gl	(156/160) 98	(156/206) 76	85	(148/153) 97	(148/205) 72	83
gr	(561/780) 72	(561/766) 73	73	(505/699) 72	(505/713) 71	72
measured	(10/12) 83	(10/15) 67	74	(8/10) 80	(8/12) 67	73
mo	(3408/5074) 67	(3408/5166) 66	67	(3008/4479) 67	(3008/4760) 63	65
mod	(3/30) 10	(3/83) 4	5	(3/27) 11	(3/67) 4	6
name_mod	(302/401) 75	(302/330) 92	83	(278/388) 72	(278/312) 89	79
number	(220/345) 64	(220/355) 62	63	(182/302) 60	(182/331) 55	58
oa	(608/837) 73	(608/810) 75	74	(579/745) 78	(579/744) 78	78
obj	(2180/2641) 83	(2180/2575) 85	84	(1961/2435) 81	(1961/2387) 82	81
oc_fin	(97/146) 66	(97/160) 61	63	(82/129) 64	(82/144) 57	60
oc_inf	(287/390) 74	(287/352) 82	77	(255/342) 75	(255/313) 81	78
og	(0/0) 0	(0/6) 0	0	(1/3) 33	(1/6) 17	22
op	(345/471) 73	(345/533) 65	69	(301/428) 70	(301/496) 61	65
part_form	(0/135) 0	(0/0) 0	0	(0/130) 0	(0/0) 0	0
pass_asp	(235/260) 90	(235/276) 85	88	(199/225) 88	(199/259) 77	82
pd	(130/226) 58	(130/295) 44	50	(115/194) 59	(115/271) 42	49
perf	(220/229) 96	(220/253) 87	91	(193/201) 96	(193/227) 85	90
precoord_form	(0/8) 0	(0/7) 0	0	(0/4) 0	(0/5) 0	0
pred_restr	(0/7) 0	(0/1) 0	0	(0/10) 0	(0/1) 0	0
pron_form	(32/32) 100	(32/40) 80	89	(29/29) 100	(29/36) 81	89
pron_type	(524/795) 66	(524/856) 61	63	(470/727) 65	(470/787) 60	62
quant	(108/184) 59	(108/158) 68	63	(105/183) 57	(105/148) 71	63
rc	(61/165) 37	(61/174) 35	36	(63/122) 52	(63/158) 40	45
rs	(0/0) 0	(0/1) 0	0	(0/0) 0	(0/1) 0	0
sb	(1798/2442) 74	(1798/2484) 72	73	(1658/2210) 75	(1658/2290) 72	74
sbp	(24/41) 59	(24/52) 46	52	(27/39) 69	(27/53) 51	59
	(19986/25986)	(19986/25721)		(18184/23534)	(18184/23767)	
TOTAL	**76.9**	**77.7**	**77.3**	**77.3**	**76.5**	**76.9**

Table 11.1: F-scores for F-structure annotation on Berkeley parser output with (split) and without (raised) LDDs resolved (TiGerDB development set) trained on TiGer48000

(test set), and the annotation of analytic future tense with *werden* improve from 90% to 92% (development set) and from 83% to 90% (test set). The annotation of coordination forms also shows an improvement, due to better recall: F-scores increase from 92% to 93% for the development set and from 88% to 91% for the test set.

I was not able to replicate Boyd (2007)'s improvement for subjects and accusative objects using the Berkeley parser. On the TiGer DB development set, the F-score for subjects (sb) increased from 73% to 74%, and for accusative objects (oa) from 74% to 78%. On the test set, however, F-scores for the split-node conversion show a decrease of 4% for subjects (sb), and no improvement for accusative objects (oa). Note that the split-node conversion yields higher precision for oa (69% (berk.split) vs. 67% (berk.raised)) but lower recall (65% (berk.split) vs. 67% (berk.raised)).

The split-node method for converting the TiGer trees to CFG representations works well for "pure" PCFG parsers like BitPar and LoPar (Boyd, 2007), where only those rules are used for parsing which have been seen in the training data. Unfortunately, parsing results for BitPar are around 10% (`evalb` labelled F-score) lower than results for the Berkeley parser or the Stanford parser, and results for a dependency-based evaluation (Kübler et al., 2008) also show that the two parsers which apply Markovisation and treebank-refinement techniques outperform "pure" PCFG parsers like BitPar and LoPar by a large margin. As mentioned before, however, both the Stanford and the Berkeley parser have considerable problems when parsing partial nodes. As CFG rules are broken up under Markovisation and new rules are generated, split nodes are often incomplete, with one partial node missing in the parser output (i.e. Markovisation may lose one or the other of the split nodes). Due to the incomplete representation of partial nodes in the parser output, the original attachment in the tree cannot be recovered. This results in lower recall scores for the split-node conversion.

While in theory the TüBa-D/Z annotation as well as the improved conversion method of Boyd (2007) for TiGer provide a means to recover LDDs in the parser output, the quality of the actual parser output trees is not good enough to successfully resolve LDDs in the trees. Currently, the automatic annotation algorithm applied to parser output from grammars extracted from the raised-node converted Tiger treebank yields better overall F-structures (evaluated against the

GF	berk.raised			berk.split		
	prec.	rec.	f-sc.	prec.	rec.	f-sc.
ams	(0/4) 0	(0/1) 0	0	(0/3) 0	(0/1) 0	0
app	(51/111) 46	(51/85) 60	52	(38/101) 38	(38/74) 51	43
app_cl	(1/19) 5	(1/17) 6	6	(1/8) 12	(1/14) 7	9
cc	(1/16) 6	(1/18) 6	6	(1/10) 10	(1/13) 8	9
circ_form	(1/1) 100	(1/1) 100	100	(1/2) 50	(1/1) 100	67
cj	(328/448) 73	(328/478) 69	71	(280/398) 70	(280/407) 69	70
comp_form	(43/46) 93	(43/50) 86	90	(33/38) 87	(33/42) 79	82
coord_form	(152/165) 92	(152/179) 85	88	(139/152) 91	(139/154) 90	91
da	(21/40) 52	(21/52) 40	46	(19/34) 56	(19/40) 48	51
det	(931/981) 95	(931/1023) 91	93	(813/864) 94	(813/913) 89	92
det_type	(957/984) 97	(957/998) 96	97	(852/878) 97	(852/892) 96	96
fut	(22/26) 85	(22/27) 81	83	(18/21) 86	(18/19) 95	90
gl	(62/68) 91	(62/81) 77	83	(46/52) 88	(46/67) 69	77
gr	(141/193) 73	(141/207) 68	70	(131/186) 70	(131/191) 69	69
measured	(3/3) 100	(3/6) 50	67	(3/3) 100	(3/5) 60	75
mo	(1130/1823) 62	(1130/1773) 64	63	(970/1592) 61	(970/1559) 62	62
mod	(0/7) 0	(0/8) 0	0	(0/10) 0	(0/8) 0	0
name_mod	(78/101) 77	(78/87) 90	83	(64/94) 68	(64/71) 90	78
number	(76/121) 63	(76/118) 64	64	(71/116) 61	(71/113) 63	62
oa	(238/355) 67	(238/356) 67	67	(197/286) 69	(197/302) 65	67
obj	(736/896) 82	(736/875) 84	83	(626/781) 80	(626/771) 81	81
oc_fin	(27/58) 47	(27/61) 44	45	(24/49) 49	(24/47) 51	50
oc_inf	(94/129) 73	(94/114) 82	77	(70/106) 66	(70/94) 74	70
og	(0/0) 0	(0/2) 0	0	(0/0) 0	(0/2) 0	0
op	(83/134) 62	(83/231) 36	45	(63/108) 58	(63/202) 31	41
part_form	(0/53) 0	(0/0) 0	0	(0/46) 0	(0/0) 0	0
pass_asp	(66/73) 90	(66/78) 85	87	(61/64) 95	(61/74) 82	88
pd	(51/95) 54	(51/112) 46	49	(48/83) 58	(48/105) 46	51
perf	(77/80) 96	(77/92) 84	90	(68/69) 99	(68/78) 87	93
postcoord_form	(0/5) 0	(0/0) 0	0	(0/3) 0	(0/0) 0	0
precoord_form	(0/3) 0	(0/4) 0	0	(0/2) 0	(0/3) 0	0
pred_restr	(0/8) 0	(0/0) 0	0	(0/3) 0	(0/1) 0	0
pron_form	(19/20) 95	(19/22) 86	90	(13/13) 100	(13/15) 87	93
pron_type	(296/401) 74	(296/446) 66	70	(221/313) 71	(221/352) 63	66
quant	(49/81) 60	(49/77) 64	62	(45/83) 54	(45/71) 63	58
rc	(20/55) 36	(20/65) 31	33	(22/38) 58	(22/58) 38	46
rs	(0/0) 0	(0/1) 0	0	(0/0) 0	(0/1) 0	0
sb	(689/923) 75	(689/943) 73	74	(567/790) 72	(567/819) 69	70
sbp	(9/11) 82	(9/14) 64	72	(6/8) 75	(6/12) 50	60
	(6789/9078)	(6789/9076)		(5830/7907)	(5830/7936)	
TOTAL	**74.8**	**74.8**	**74.8**	**73.7**	**73.5**	**73.6**

Table 11.2: F-scores for F-structure annotation on Berkeley parser output with (split) and without (raised) LDDs resolved (TiGerDB test set)

TiGer DB test set where LDDs are resolved) than the ones generated in the other settings.

11.3 Improving Coverage with SubCat Frames

So far I have presented different architectures for treebank-based LFG grammar acquisition and parsing for German. Some of the approaches achieve quite good results for precision, but recall is still a serious problem. Especially for the two-step model, where I train the Berkeley parser on syntactic nodes only and assign the grammatical functions in a post-processing step, missing context sensitivity of the function labeller leads to clashes in the constraint solver when resolving the F-structure equations. Many of these clashes are caused by the presence of more than one governable grammatical function of the same type in the same local tree. Below I describe an attempt to solve this problem and to disambiguate grammatical function labels with the help of automatically extracted subcategorisation frames.

I automatically extract subcategorisation frames from the TiGer treebank to resolve ambiguities when the same governable grammatical function appears twice in the same local tree. Figure 11.1 shows a parser output tree from the TiGer DB development set where FunTag annotated both the sentence-initial NP as well as the personal pronoun with the subject label. Both nodes are, in fact, probable candidates for the subject role: the NP because of its sentence-initial position, the personal pronoun due to its property of being animate. The word form of the determiner, which, for humans, identifies the NP as a dative object, does not have enough weight to influence the decision of FunTag, probably due to sparse data.

Subcat frame information can help to disambiguate cases like the one above (Figure 11.1). The idea is quite simple: if we know the most probable subcategorisation frame for the head verb of the sentence, we can assign grammatical functions to nodes in the tree according to the subcat frame.

To be able to do this, we need subcategorisation frames for all verbs in the treebank. I automatically extract these frames from the F-structure-annotated treebanks, which encode all governable functions for each predicate and allow us to compute the probability for each particular subcat frame.

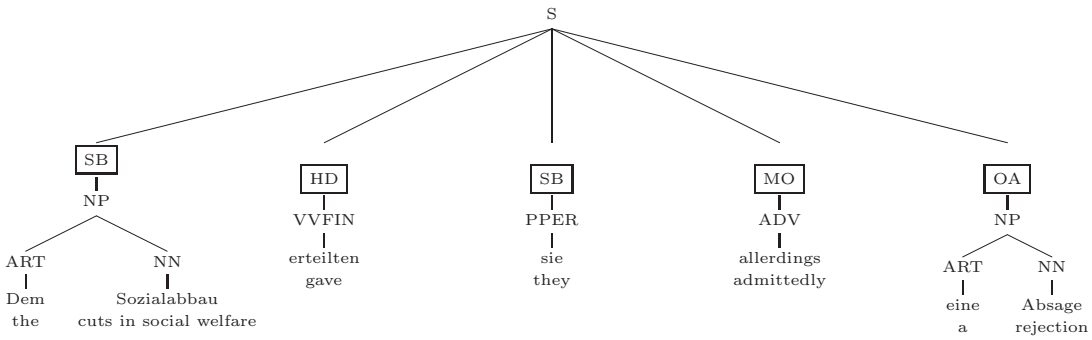


Figure 11.1: FunTag error: the same GF (SB) appearing twice in the same local tree

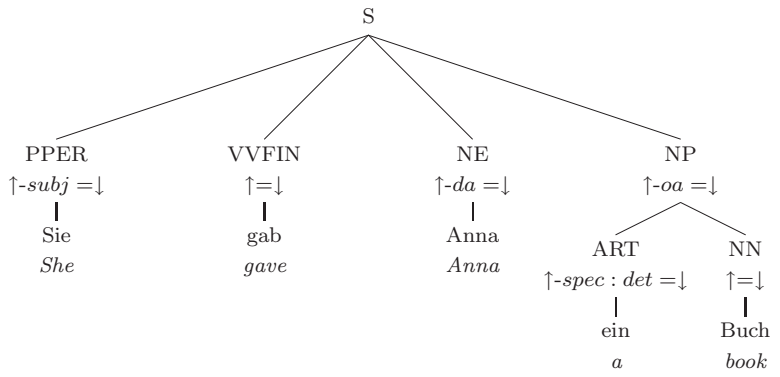
11.3.1 SubCat Frame Extraction

In my subcat frame extraction experiments I follow O'Donovan et al. (2004) and O'Donovan et al. (2005a), who describe the large-scale induction and evaluation of lexical resources from the Penn-II and Penn-III treebanks.

O'Donovan et al. extract grammatical syntactic-function-based subcategorisation frames (LFG semantic forms) as well as traditional CFG category-based subcategorisation frames with varying degrees of detail. They extract subcat frames with and without subcategorised PPs, and they are able to specify the syntactic category of a subcategorised grammatical function. Furthermore, they distinguish between active and passive frames, which crucially improves the quality of the induced resources. In contrast to other approaches, the method of O'Donovan et al. does not predefine the number and type of the frames to be induced.

O'Donovan et al. associate probabilities with frames, conditioned on the lemma form of the predicate. Most important, the induced frames fully reflect non-local dependencies in the data, which makes them a truly deep linguistic resource.

I apply the method of O'Donovan et al. (2004, 2005a) to the German treebanks and acquire LFG semantic forms from the automatically F-structure-annotated TiGer and TüBa-D/Z treebanks.



Sie gab Anna ein Buch
She gave a book to Anna

$$\begin{bmatrix} \text{PRED} & \text{'geben} \left\langle \text{subj, da, oa} \right\rangle\text{'} \\ \text{SUBJ} & \left[\text{PRED} \quad \text{pro}\right] \\ \text{DA} & \left[\text{PRED} \quad \text{'Anna'}\right] \\ \text{OA} & \begin{bmatrix} \text{SPEC:DET:PRED} & \text{'ein'} \\ \text{PRED} & \text{'Buch'} \end{bmatrix} \end{bmatrix}$$

Figure 11.2: LFG c-structure and F-structure

SubCat Frame Extraction: Methodology

In order to be able to extract verb frames from the two treebanks, I first annotated the treebanks with LFG F-structure equations, using the automatic annotation algorithm described in Section 10.2. In my experiments I use two different data sets. In order to support a meaningful comparison of subcat frames induced from TiGer and TüBa-D/Z, I annotated the two training sets with 25 000 sentences each, as used in the parsing experiments in Chapter 10. For TiGer, I also repeated the experiment with the larger training set with 48,473 sentences. The set of semantic forms extracted from the large training set was then used for resolving ambiguities caused by duplicate governable function labels.

After annotating the data with LFG F-structure equations and producing the F-structures using a constraint solver, the subcategorisation frame extraction algorithm traverses each F-structure f and, for each predicate in f, collects all governable grammatical functions on the same level. For the tree in Figure 11.2 and its corresponding F-structure I extract the following LFG semantic form (11.1).

$$sf(geben([subj, da, oa]) \tag{11.1}$$

Including syntactic information from the CFG tree into the subcategorisation frame results in frame (11.2). I distinguish four different POS tags: verbs (v), nouns (n), prepositions (p) and pronouns (pro).

$$sf(geben([subj(pro), da(n), oa(n)]) \tag{11.2}$$

The frames can be refined by including additional information like subcategorised PPs (11.3) or by specifying the form of a complementiser (11.4).

$$sf(stellen([subj(n), oa(n), pp(auf)]) \tag{11.3}$$

$$sf(sagen([subj(n), comp(dass)]) \tag{11.4}$$

The set of grammatical features used in the annotation algorithm also allows us to distinguish between syntactic and semantic roles, as in the diathesis alternation, where the syntactic subject in the active verb frame corresponds to the semantic role often referred to as Agent, while for passive voice the syntactic subject corresponds to a semantic role often expressed as Theme, Patient or Experiencer (Examples 38,39).

$$sf(braten([subj(n), oa(n)]) \tag{11.5}$$

(38) Anna$_{AGENT}$ brät einen Storch$_{THEME}$
Anna fries a stork
Anna is frying a stork

$$sf(braten([subj(n)], passive : stativ) \tag{11.6}$$

(39) Der Storch$_{THEME}$ ist gebraten
The stork is fried
The stork has been fried

The *passive* : *stative* in Example (11.6) identifies the frame as a stative passive verb frame, adding the information which is crucial for the correct semantic interpretation of the whole expression. The F-structure annotations allow us to distinguish between different passive aspects like stative passive, dynamic passive or modal passive.

Semantic Forms for TiGer and TüBa-D/Z

Depending on the granularity of the subcat frame extraction, I extract the subcategorisation frames in Table 11.3 for the TiGer training set (25,000 sentences), the TüBa-D/Z training set (25,000 sentences) and for the large TiGer training set (48,473 sentences).

	lemma types	**gf**	**gf(POS)**	**gf(POS), pp**	**gf(POS), pp passive, comp**	
TüBa-D/Z	2638	6999	10202	10894	11489	*verb*
(25 000)	106	107	231	231	231	*prep*
TiGer	3434	8514	12644	12810	14002	*verb*
(25 000)	103	141	280	280	284	*prep*
TiGer	4590	12170	19085	19389	21582	*verb*
(48 000)	118	179	353	353	359	*prep*

Table 11.3: Subcat frame types for verbs and prepositions for TiGer and TüBa-D/Z; gf=grammatical functions; gf(POS)=gf + POS/syntactic information; gf(POS),pp=including prepositions; gf(POS),pp,passive,comp=including voice and comp-form

The first column shows the number of different lemma types in the data sets. We observe a far higher number of different verb types in the TiGer treebank

than in the TüBa-D/Z, which is consistent with the difference in vocabulary size reported in Section 5.2. For the closed word class of prepositions the frequencies are quite close, with 106 vs. 103 in TüBa-D/Z and TiGer (25,000 sentences). For the large data set (TiGer) the number is slightly higher with 118 different types, while for the open word class of verbs the number of lemma types increases considerably to 4590 for the full TiGer set.

The next four columns report the number of subcategorisation frame types extracted from the treebanks for different degrees of information. The more fine-grained the information encoded in the semantic forms, the higher the number of different frame types we extract. For all four classes (1: grammatical functions (gf), 2: gf with syntactic information (gf(POS)), 3: gf(POS) with prepositions (gf(POS), pp), 4: gf(POS), pp, including passive voice and word form of complementiser (gf(POS), pp, passive, comp)), the number of frame types extracted from TiGer is significantly higher than the one extracted from the TüBa-D/Z. As discussed in Section 5.2, there are two possible reasons for this: stylistic differences between the two newpapers as well as the length of time period covered by the articles, which influences the variety of topics and also the number of *hapax legomena* (which often are names of persons, institutions or locations) in the newspaper text.

11.3.2 Using SubCat Frames for Disambiguation

The motivation for extracting the subcat frames is based on the idea to use them to correct erroneously function-labelled parse trees, where the parser or the function labeller assigned incorrect (here duplicate) grammatical function labels, causing clashes when resolving the F-structure equations.

I proceed as follows: the tree in Figure 11.1 would give us the (erroneous) subcategorisation frame in (11.7), where we have a subject NP (subj(n)) and a personal pronoun also bearing the subject label (subj(pro)).

$$sf(erteilen([subj(n), subj(pro), oa(n)]) \tag{11.7}$$

In order to correct the analysis and generate an F-structure for this tree, one of the duplicate grammatical functions has to be changed. I automatically generate regular expressions describing all possible solutions for resolving the conflict (11.8,

11.9, and 11.10).[23]

$$sf(erteilen([X(n), subj(pro), oa(n)]) \tag{11.8}$$

$$sf(erteilen([subj(n), X(pro), oa(n)]) \tag{11.9}$$

$$sf(erteilen([subj(n), subj(pro), X(n)]) \tag{11.10}$$

Next I retrieve the automatically extracted subcat frames for the lemma *erteilen* (Table 11.4) from the F-structure-annotated TiGer or TüBa-D/Z, as required. I consider all subcat frames with the same number of arguments as in the erroneous form (11.7). Let us assume we extracted 10 different subcat frames for *erteilen*, out of which three frames have three arguments (Table 11.4). Out of these three subcat frames, we are looking for one with an NP or a noun as first argument, followed by a pronoun, and again an NP/noun as its last argument. Note that the arguments in the subcat frames are ordered according to their position in the surface string, in order to capture preferences like realising the subject in a sentence-initial position. Only one out of the three subcat frames meets these requirements, and this is the one giving us the correct grammatical function assignment *(da(n), sb(pro), oa(n))* for the example under consideration. In cases where there is more than one matching frame, the frame with the highest probability is chosen.

Following this method, the annotation algorithm tries to validate all parser output trees with conflicting grammatical functions and to assign the correct function labels according to subcat frame information, ranked according to their probability conditioned on the lemma form.

Results for SubCat Frame-Based Disambiguation

Table 11.5 shows F-structure evaluation results for the subcat frame-based disambiguation method trained on TiGer and using the TiGer DB dependency gold standard. I applied the approach to the parser output of the Berkeley parser (berk.par) and to the output of the two-step architecture (berk.fun, berk.fun.par)

[23] (11.10) is not correct, either, but we can be sure that there will be no subcat frame from the F-structure-annotated treebanks matching this template.

lemma form		*arguments*				*probability*
$sf(erteilen$	$([$	$da(n),$	$sb(pro),$	$oa(n)$	$]),$	0.037037037037037).
$sf(erteilen$	$([$	$sb(n)$			$]),$	0.037037037037037).
$sf(erteilen$	$([$	$sb(pro)),$	$oa(n)$		$]),$	0.037037037037037).
$sf(erteilen$	$([$	$da(n),$	$sb(n),$	$oa(n)$	$]),$	0.222222222222222).
$sf(erteilen$	$([$	$oa(n),$	$sb(pro)$		$]),$	0.148148148148148).
$sf(erteilen$	$([$	$da(n),$	$sb(n)$		$]),$	0.037037037037037).
$sf(erteilen$	$([$	$sb(n),$	$da(n),$	$oa(n)$	$]),$	0.259259259259259).
$sf(erteilen$	$([$	$sb(n),$	$oa(n)$		$]),$	0.111111111111111).
$sf(erteilen$	$([$	$oa(n)$			$]),$	0.0740740740740741).
$sf(erteilen$	$([$	$sb(n),$	$da(n)$		$]),$	0.037037037037037).

Table 11.4: Automatically extracted subcat frames for *erteilen* (to give, to grant)

and evaluated the resulting F-structures against the TiGer DB development and test set.

For all three parser settings (berk.par, berk.fun, berk.fun.par) there is a slight decrease in F-score when applying the subcat frame disambiguation method. Coverage, however, increases considerably. The gain is more profound for the FunTag architecture, where we achieve up to 5% absolute increased F-structure coverage. The disambiguation method does improve coverage, but there still remain about 10-15% of the sentences which cannot be resolved into a F-structure.

This means that the coverage of our automatically extracted subcat frames is not yet good enough. Table 11.6 shows the number of GF label conflicts in the parser/FunTag output trees, and also the number of conflicts for which we found a disambiguating subcat frame. The coverage problem might also be due to the fact that I encoded the surface position of the arguments in a sentence into the subcat frames. This produces very precise subcategorisation frames, but at the cost of coverage and sparse data. To overcome the problem I implemented a back-off method, where for cases where the system does not find a linearised subcat frame, I permute the arguments in the frame and test all possible combinations in order to find a matching subcat frame. Table 11.7 shows results for the subcat frame-based disambiguation with back-off. Precision and recall are more or less the same as in Table 11.5, while the number of resolved conflicts in the FunTag output

	precision	recall	F-score	% valid F-structures
TiGer48000 - F-structure evaluation - development set				
berk.par	77.7	78.3	78.0	88.5%
berk.par.sf	77.0	77.9	77.4	91.2%
berk.fun	78.9	71.1	74.8	88.4%
berk.fun.sf	78.3	70.8	74.4	93.0%
berk.fun.par	78.3	68.0	72.7	85.4%
berk.fun.par.sf	77.4	67.6	72.1	90.5%
TiGer48000 - F-structure evaluation - test set				
berk.par	76.0	76.5	76.2	84.2%
berk.par.sf	74.8	75.9	75.3	86.2%
berk.fun	76.5	66.7	71.3	84.2%
berk.fun.sf	76.0	66.9	71.1	88.2%
berk.fun.par	76.3	61.7	68.2	83.4%
berk.fun.par.sf	75.4	60.3	67.0	88.8%

Table 11.5: F-structure evaluation results for subcat frame-based disambiguation method on the TiGerDB

	# GF conflicts	*sf*	*sf + back-off*
berk.par	95	40	46
berk.fun	160	87	95
berk.fun.par	172	94	99

Table 11.6: Number of conflicting GF labels and number of matching subcat frames without and with back-off (TiGer DB development and test set) trained on TiGer48000

	precision	**recall**	**F-score**	**% valid F-structures**
TiGer48000 - F-structure evaluation - test set				
berk.par.sf	74.8	75.9	75.3	86.2%
+ back-off	74.8	75.9	75.3	86.2%
berk.fun.sf	76.0	66.9	71.1	88.2%
+ back-off	75.8	66.7	71.0	88.4%
berk.fun.par.sf	75.4	60.3	67.0	88.8%
+ back-off	75.4	60.3	67.0	89.2%

Table 11.7: F-structure evaluation results for the subcat frame-based disambiguation method + back-off for the TiGer DB

increases further (Table 11.6), as does the number of F-structures. For the parser-assigned grammatical functions we do not observe any further improvement.

For the Berkeley parser-assigned grammatical function labels, a total of 46 GF conflicts could be solved using linearised subcat frames plus the back-off method, while for the remaining 49 cases no matching subcat frame was found (Table 11.6). In the gold standard-trained FunTag output, we found 160 conflicting grammatical function labels, 95 of which could be solved, while in the parser output-trained FunTag setting the number of conflicting GF labels was higher at 172, as was the number of cases where the conflict could be solved (99) by applying the subcat frame-based method.

The subcat frame-based approach to improve F-structure coverage with the help of automatically extracted subcat frames yields an absolute improvement of

up to 5% more valid F-structures. However, Table 11.6 also shows that for nearly half of the incorrectly labelled trees, no matching subcat frame could be found. This means that the TiGer treebank is not large enough as a resource for subcat frame extraction to yield sufficient coverage.

11.4 Conclusions

This chapter presented two extensions to the F-structure annotation algorithm for German:

1. the generation of proper F-structures for the TiGer treebank, based on Boyd (2007)'s split-node conversion method to recover LDDs in the parser output;

2. a method to improve coverage, based on automatically extracted subcategorisation frames.

The proper F-structures with LDDs resolved show better results for some of the dependencies included in the F-structure evaluation, while overall results are slightly higher for F-structures generated from parser output of the Berkeley parser trained on the "shallow" raised-node version of the TiGer treebank. The main problem for recovering LDDs is caused by incomplete representations of partial nodes in Markovisation-based parser output (Berkeley parser). This means that the original tree structure cannot be reconstructed, which results in lower recall for F-structures generated from berk.split parser output as well as in incorrect F-structure analyses. A possible solution to this problem might consist of a preprocessing step, where parser output trees with incomplete partial node representations are mapped against tree structures from the original split-node-converted treebank, and the corrupted trees are corrected. The mapping process, however, is not straightforward. For each partial node in the parser output missing its corresponding split node, we have to decide whether a second partial node should be inserted, or whether we should delete the single partial node from the parser output tree. In the first case, we have to find a grammar rule in the gold trees which can be mapped to the grammar rule for the erroneous parser output tree. Due to the flat tree structure in TiGer, which result in many low-frequency

rules, we might not be able to find a fitting rule, and further generalisations over the actual tree structure are necessary. This comes at the risk of introducing more noise into the trees.

The second extension presented in this chapter describes a method for improving coverage based on subcategorisation frames bootstrapped from the F-structure-annotated TiGer treebank. The method achieves an improvement in coverage of more than 5% on the output of the two-step architecture (evaluated against the TiGer DB test set), and a less profound improvement of 2% for F-structures generated in the one-step architecture. While these results are promising, the error analysis showed that the method still suffers from sparse data: for half of the incorrectly labelled tree structures in the parser output no matching subcat frame could be found. This means that including a larger subcat frame resource might further improve coverage.

Chapter 12

Parsing: Related Work

12.1 Introduction

The last four chapters described the substantially extended and improved acquisition of deep, wide-coverage LFG resources for German (Chapters 8,9) and presented parsing architectures and experiments parsing German into LFG F-structures (Chapters 10,11). This chapter discusses related work and shows how my research compares to a wide-coverage hand-crafted LFG grammar (Dipper, 2003; Rohrer and Forst, 2006; Forst, 2007).

12.2 Related Work

The only other broad-coverage LFG grammar for German I am aware of is the hand-crafted LFG (Dipper, 2003; Rohrer and Forst, 2006; Forst, 2007) developed in the ParGram project (Butt et al., 2002). The ParGram German LFG uses 274 LFG-style rules (with regular expression-based right-hand sides) and several lexicons with detailed subcategorisation information and a guessing mechanism for default lexical entries (Rohrer and Forst, 2006). Preprocessing in the experiments reported in Rohrer and Forst (2006) includes modules for tokenisation, morphological analysis and manual marking of named entities, before the actual parsing takes place. An additional disambiguation component based on maximum entropy models is used for reranking the output of the parser. Forst (2007) tested parser quality on 1497 sentences from the TiGer DB and reported a lower bound,

	ParGram			TiGerDB		DCU250		CCG2000
GF	up. bound	log. lin.	low. bound	raised	raised + sf	raised	raised + sf	raised + sf DCU250-style
da	67	63	55	44	45	38	35	41
gr	88	84	79	71	70	87	87	87
mo	70	63	62	65	63	73	72	72
oa	78	75	65	69	68	63	61	70
quant	70	68	67	67	64	78	78	88
rc	74	62	59	34	32	30	28	44
sb	76	73	68	74	74	79	80	83
preds only	79.4	75.7	72.6	72.7	71.5	78.6	77.9	80.9
	coverage on the NEGRA treebank (>20,000 sentences)							
	81.5	81.5	81.5	88.2	89.5	88.7	89.9	89.9

Table 12.1: F-scores for selected grammatical functions for the ParGram LFG (upper bounds, log-linear disambiguation model, lower bounds) and for the TiGer grammars (berk.par)

where a parse tree is chosen randomly from the parse forest, an upper bound, using the parse tree with the highest F-score (evaluated against the gold standard), as well as results for parse selection done by the log-linear disambiguation model.

Table 12.1 shows results for the ParGram LFG and for the automatically induced grammars on selected grammatical relations and on all grammatical functions excluding morphological and other features (preds only). The automatically induced TiGer DB and DCU250-style grammars were trained on the full TiGer treebank (>48,000 sentences, excluding the test data), while the CCG2000-style grammar was trained on the 25,000 sentences training set. I report results for the test sets from the TiGer DB, the DCU250 and the CCG2000.

The hand-crafted LFG outperforms the automatically induced grammars on most GFs for the TiGer DB, but results are not directly comparable. The TiGer DB-based evaluation is biased in favour of the hand-crafted LFG. Named entities in the ParGram LFG input are marked up manually, while for our grammars these multiword units often are not recognised correctly and so are punished during evaluation, even if part of the unit is annotated correctly. Furthermore,

the hand-crafted ParGram LFG grammar was used in the creation of the TiGer DB gold standard in the first place, ensuring compatibility as regards tokenisation and overall linguistic analysis.

F-scores for the DCU250 are in roughly the same range as the ones for the hand-crafted grammar. For high-frequency dependencies like subjects (sb) or modifiers (mo), results of the two grammars are comparable. For low-frequency dependencies like dative objects (da) or relative clauses (rc), however, the hand-crafted LFG outperforms the automatic LFG F-structure annotation algorithm by far. Coverage for the automatically induced grammars is considerably higher than for the hand-crafted LFG grammar. Rohrer and Forst (2006) report a coverage of 81.5% (full parses) when parsing the NEGRA treebank, which contains newspaper text from the same newspaper as in the TiGer treebank. By contrast, the automatically induced TiGer grammars achieve close to 90% coverage on the same data. On the TiGer treebank Rohrer and Forst (2006) report coverage of 86.44% full parses, raising the possibility that, as an effect of enhancing grammar coverage by systematically extracting development subsets from TiGer, the ParGram LFG is tailored closely to the TiGer treebank.

The CCG2000 test set is equally biased towards the TiGer treebank-based LFG resources, as it only represents what is encoded in the automatic F-structure annotation algorithm. The best F-structure parsing results, 81.9% F-score for the hand-crafted ParGram LFG against TiGer DB and the 80.9% F-score against the CCG2000 for the treebank-based LFG, clearly show the bias. The truth is somewhere in between: The TiGer DB evaluation of the treebank-based LFG resources attempts to a limited extend to counter the bias of the original TiGer DB resource towards the hand-crafted LFG grammar by removing distinctions which cannot be learned from TiGer data only, and by relating TiGer DB to (some of) the original TiGer tokenisation using the version prepared by Boyd et al. (2007). The resulting resource still favours the hand-crafted LFG resources, which outperform the treebank-based resources by about 5% points absolute.

12.3 Discussion

Our automatically extracted grammars yield better coverage than the hand-crafted LFG of (Dipper, 2003; Rohrer and Forst, 2006; Forst, 2007), but with

	ParGram			TiGerDB		DCU250		CCG2000
	up.	log.	low.	raised	raised	raised	raised	raised + sf
GF	bound	lin.	bound		+ sf		+ sf	DCU250-style
	F-score			*precision*				
da	67	63	55	58	54	50	57	68
gr	88	84	79	68	68	88	88	87
mo	70	63	62	63	62	77	76	75
oa	78	75	65	68	71	80	82	74
quant	70	68	67	58	56	69	69	91
rc	74	62	59	50	49	50	50	48
sb	76	73	68	76	77	84	87	88
preds only	83.3	76.2	73.7	76.0	83.7	84.4	85.4	85.5

Table 12.2: Precision for selected grammatical functions for the ParGram LFG and for the TiGer grammars (two-step architecture; berk.fun)

regard to F-score the ParGram LFG still outperforms the automatically acquired grammars. The lower results for our grammars are not due to low precision: Table 12.2 contrasts F-scores for the Pargram LFG with results for precision as achieved by the automatically acquired TiGer grammars (two-step architecture, berk.fun).[24] Future work should therefore focus on improving recall in order to achieve results comparable with or better than hand-crafted grammars.

In Chapter 11 I showed that recall for the two-step architecture can be improved using subcategorisation frames automatically extracted from the TiGer treebank. However, the TiGer treebank is not large enough as a resource for subcat frame extraction. Subcat frames automatically induced from a larger data set might provide further improvements.

Another unsolved problem is the encoding of LDDs in treebank annotation schemes for (semi-)free word order languages. Currently, neither the TiGer treebank and even less so the TüBa-D/Z way of representing non-local dependencies can be learned successfully by statistical parsers. An approach to resolving LDDs on F-structure level was described in Section 7.1.5 and successfully implemented as part of the English treebank-based LFG acquisition and parsing architectures

[24] Unfortunately, Forst (2007) does not report results for precision and recall.

(Cahill et al., 2004; Cahill, 2004). However, the method of Cahill et al. relies on complete F-structures, which means that the recall problem must have been solved before we can reliably and profitably compute LDDs on F-structure level for German.

Chapter 13

Conclusions

Automatic acquisition of deep, wide-coverage linguistic resources is of great importance for many areas of NLP. Successful lines of research have been presented for the automatic acquisition of rich and deep resources for English and the Penn-II treebank, but so far it has not been clear whether these approaches are as successful when applied to other languages with linguistic characteristics substantially different from English and treebanks with data structures and encoding conventions different from the Penn treebanks.

In this thesis I address these questions and present a thorough comparison of two German treebanks with different annotation schemes. I investigate the impact of language-specific properties and treebank-specific data structures on PCFG parsing and data-driven LFG grammar acquisition. Below I summarise my main findings.

13.1 Is German Harder to Parse than English?

In Chapter 4 I show that the claim that German is not harder to parse than English (Kübler, 2005; Kübler et al., 2006; Maier, 2006) does not hold. I present controlled error insertion experiments showing that the PARSEVAL metric is not a valid evaluation measure for cross-treebank comparisons and that it does not fully reflect parser output quality in a linguistically adequate way. More evidence for the inadequacy of PARSEVAL was presented in Chapter 6, where we show that constituency-based parsing results do not necessarily correlate with results

of a dependency-based evaluation, the latter being more suitable to capture linguistically relevant information like predicate-argument structure. Results from a manual evaluation on a testsuite with complex German grammatical constructions, the TePaCoC, reinforce the findings from the dependency-based evaluation. Even more evidence comes from the evaluation of automatically annotated LFG F-structures in Chapter 10, where again there was no consistent agreement between constituency-based parsing results and results for LFG F-structures, representing functional dependency relations.

13.2 Comparing Treebank Design - TiGer and TüBa-D/Z

The question of whether German is harder to parse than English or not is not yet decided. However, semi-free word order together with case syncretism increases structural ambiguity and poses a great challenge for the design of treebanks. I investigate the question as to which of the annotation schemes of the two German treebanks, TiGer and TüBa-D/Z, is more suitable for PCFG parsing and for the automatic acquisition of deep, wide-coverage LFG resources. In Chapter 5 I discuss methodological problems arising for cross-treebank comparisons. Chapter 6 presents a way to compare PCFG parser performance for parsers trained on treebanks as different as the TiGer treebank and the TüBa-D/Z. Results from a labelled dependency-based evaluation provides evidence that the flat annotation in TiGer is more transparent and so compensates for the high number of long, low-frequency rules. These results are backed up by a manual evaluation of a carefully selected testsuite, the TePaCoC, containing sentences with complex grammatical constructions from each of the treebanks. The testsuite allows us to detect error types and trace them back to the treebank annotation decision underlying the error. It complements the evaluation using automatic metrics and supports a linguistically motivated assessment of parser output quality across different treebanks.

In Chapter 8 I discuss the pros and cons of specific design decisions in TiGer and TüBa-D/Z for the automatic acquisition of deep, wide-coverage LFG re-

sources. I show that the annotation in TüBa-D/Z causes several problems for the grammar acquisition task, one of them being the design of the grammatical function labels, which in the TüBa-D/Z include information about non-local dependencies in the trees. This would, in theory, allow us to generate proper LFG F-structures with LDDs resolved. My experiments, however, show that these labels are harder to learn than the grammatical function labels in TiGer, which exclusively focus on encoding functional information related to the syntactic nodes they are assigned to. The close relationship between nodes and labels makes them easy to understand for humans, and also improves their learnability for machine learning-based methods. In addition, the TüBa-D/Z labels encode less specific linguistic information than the labels in the TiGer treebank.

13.3 Is Treebank-Based Grammar Induction for German feasible?

In Chapter 10 I present approaches to acquire deep, wide-coverage LFG resources for German. In my experiments I test the performance of three parsers trained on two treebanks. I compare the impact of two methods for converting crossing branches in TiGer into CFG trees. I assess the quality of parser-assigned grammatical functions in the trees, which for German are essential for automatic F-structure annotation, and grammatical function labels learned by an SVM-based function labeler.

I provide an extensive evaluation against three hand-crafted gold standards and against a larger data set of automatically annotated dependency triples (CCG-style evaluation). Error analysis shows that precision for F-structures generated from TiGer-trained parser output is quite high, especially for the F-structures generated from the output of the SVM-based function labeler. Coverage, however, is a serious problem, reflected in low recall, especially for the SVM-based function labeling architecture. Here the local decisions made by the SVM in combination with the flat annotation in the TiGer treebank result in violations of the LFG coherence condition, due to the assignment of more than one governable grammatical function of the same type in the same local tree. I

present a method to improve coverage with the help of subcategorisation frames, automatically extracted from LFG F-structures, generated from the annotated TiGer treebank.

It is difficult to directly compare my results with the hand-crafted LFG grammar of Rohrer and Forst (2006). The automatically acquired grammars are superior with regard to coverage, and yield precision scores in the same range as the ones for the hand-crafted grammar. Comparing the overall F-scores, the hand-crafted LFG outperforms the treebank-based grammars.

13.4 Future Work

The main problems for the automatic acquisition of LFG resources for German are the following:

- the low CFG parsing results for German, especially when considering combined node and grammatical function labels;
- low recall especially for the SVM-based architecture;
- the adequate representation of LDDs in the treebank.

Improving results for syntactic parsing of German is essential for data-driven grammar acquisition, as our approach heavily relies on the grammatical function labels in the German treebanks. In order to improve results, we need to improve, or to develop new parsing techniques which can handle the high ambiguity caused by the semi-free German word order together with case syncretism. The approach of assigning GF labels in a post-processing step, using an SVM-based function labeler, showed promising results. However, the gain in precision was paid at the cost of an unacceptable decrease in recall. The SVM classifier treats the problem as a binary classification task, treating each GF label on its own. Future work should investigate joint models for the assignment of grammatical functions, in order to prevent conflicts between multiple subjects or objects assigned to the same local tree.

Another possible line of research could look into the feature sets used to train the SVM. These features claim to be language-independent (Chrupala et al.,

2007) and have been used successfully to assign grammatical function labels to the English Penn-II treebank (Bies et al., 1995), the Spanish Cast3LB treebank (Civit and Marti, 2004) as well as the Penn Chinese treebank (Xue et al., 2005). However, I do believe that language-dependent as well as treebank-dependent feature tuning could substantially improve the method, as it cannot be expected that the same extraction method will capture all relevant clues for all treebank encoding schemes and for typologically different languages.

The most challenging problem consists of an appropriate representation of non-local dependencies for a semi-free word order language. The two German treebanks chose different ways to solve this problem, which both proved to be difficult for machine learning methods. The question at hand is how one can identify and encode features which express non-local dependencies without causing a sharp increase in the number of categories that need to be learned, resulting in data sparseness, and the question whether those categories can be distinguished based on local distribution only. This problem has to be solved before we can hope to automatically acquire really high-quality deep linguistic resources for German.

References

Arun Abhishek and Frank Keller. Lexicalization in crosslinguistic probabilistic parsing: The case of french. In *43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pages 306–313, Ann Arbor, Michigan, 2005. 24, 35

Hiyan Alshawi, editor. *The Core Language Engine*. MIT Press, Cambridge, MA, 1992. 22

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. Map adaptation of stochastic grammars. *Computer Speech and Language*, 20(1):41–68, 2006. 22

Judith Berman. Topicalization vs. left dislocation of sentential arguments in german. In *Proceedings of the 1st International Lexical Functional Grammar Conference (LFG-96)*, pages 75–88, Grenoble, Suitzerland, 1996. 15

Manfred Bierwisch. Grammatik des deutschen verbs. *Studia grammatica*, 2, 1963. 15

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. *Bracketing Guidelines for Treebank II Style Penn Treebank Project*. University of Pennsylvania, 1995. 208

Ezra W. Black, Steven Abney, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Fred Jelinek, Judith Klavans, Mark Liberman, Mitch Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. A procedure for quantitatively comparing the syntactic coverage

of english grammars. In *In Proceedings of the DARPA Speech and Natural Language Workshop*, pages 306–311, San Mateo, CA, 1991. 9, 22, 23, 35, 36

Don Blaheta and Eugene Charniak. Assigning function tags to parsed text. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, San Francisco, CA, USA, 2000. 165

Rens Bod. A computational model of language performance: Data oriented parsing. In *The 14th International Conference on Computational Linguistics*, pages 855–859, Nantes, France, 1992. 22

Adriane Boyd. Discontinuity revisited: An improved conversion to context-free representations. In *Proceedings of the Linguistic Annotation Workshop (LAW 2007)*, pages 41–44, Prague, Czech Republic, 2007. x, 8, 143, 145, 146, 183, 185, 197

Adriane Boyd and Detmar Meurers. Revisiting the impact of different annotation schemes on pcfg parsing: A grammatical dependency evaluation. In *ACL Workshop on Parsing German (PaGe-08)*, pages 24–32, Columbus, OH, 2008. 59, 60

Adriane Boyd, Markus Dickinson, and Detmar Meurers. On representing dependency relations – insights from converting the german tigerdb. In *Proceedings of the 6th International Workshop on Treebanks and Linguistic Theories (TLT-07)*, pages 31–42, Bergen, Norway, 2007. 115, 120, 201

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER Treebank. In Erhard W. Hinrichs and Kiril Simov, editors, *Proceedings of the First Workshop on Treebanks and Linguistic Theories*, pages 24–42, Sozopol, Bulgaria, 2002. 14, 21

Thorsten Brants. Tnt - a statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing (ANLP)*, pages 224–231, Seattle, WA, 2000. 29

Michael R. Brent. Automatic acquisition of subcategorization frames from untagged text. In *The 29th annual meeting on Association for Computational Linguistics*, pages 209–214, Berkeley, CA, 1991. 3

Michael R. Brent. From grammar to lexicon: Unsupervised learning of lexical syntax. *Computational Linguistics*, 19(2):243–262, 1993. 3

Joan Bresnan. *Lexical-Functional Syntax*. Blackwell, Oxford, 2000. 105

Ted Briscoe and John Carroll. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-02)*, pages 1499–1504, Las Palmas, Canary Islands, 2002. 105

Ted Briscoe and John A. Carroll. Automatic extraction of subcategorization from corpora. In *Proceedings of the 5th ANLP Conference*, pages 356–363, Washington DC, 1997. 3

Ted Briscoe, Claire Grover, Bran Boguraev, and John A. Carroll. A formalism and environment for the development of a large grammar of english. In *Proceedings of the 4th ACL/SIBPARSE International Workshop on Parsing Technologies*, pages 703–708, Milan, Italy, 1987. 22

Michael Burke, Aoife Cahill, Mairéad McCarthy, Ruth O'Donovan, Josef van Genabith, and Andy Way. Evaluating automatic f-structure annotation for the penn-ii treebank. *Journal of Language and Computation; Special Issue on Treebanks and Linguistic Theories*, pages 523–547, 2004a. 110

Michael Burke, Olivia Lam, Aoife Cahill, Rowena Chan, Ruth O'Donovan, Adams Bodomo, Josef van Genabith, and Andy Way. Treebank-based acquisition of a chinese lexical-functional grammar. In *Proceedings of the 18th Pacific Asia Conference on Language, Information and Computation (PACLIC-18)*, pages 161–172, Tokyo, Japan, 2004b. 1, 4, 110

Miriam Butt, María-Eugenia Ni no, and Frédérique Segond. Multilingual processing of auxiliaries within lfg. In *Proceedings of KONVENS 1996*, pages 111–122, Bielefeld, Germany, 1996. 124

Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. The parallel grammar project. In *Proceedings of COLING-02 Workshop on Grammar Engineering and Evaluation*, Taipei, Taiwan, 2002. 110, 199

Aoife Cahill. *Parsing with Automatically Acquired, Wide-Coverage, Robust, Probabilistic LFG Approximations*. PhD dissertation, School of Computing, Dublin City University, Dublin, Ireland, 2004. 1, 4, 5, 7, 10, 24, 28, 29, 103, 104, 107, 108, 110, 111, 112, 113, 114, 115, 116, 121, 125, 133, 203

Aoife Cahill, Mairéad McCarthy, Josef van Genabith, and Andy Way. Automatic annotation of the penn-treebank with lfg f-structure information. In *LREC-02 workshop on Linguistic Knowledge Acquisition and Representation - Bootstrapping Annotated Language Data, Third International Conference on Language Resources and Evaluation (LREC-02), post-conference workshop*, pages 8–15, Paris, France, 2002. 1, 4, 104, 107, 110

Aoife Cahill, Martin Forst, Mairéad McCarthy, Ruth O' Donovan, Christian Rohrer, Josef van Genabith, and Andy Way. Treebank-based multilingual unification-grammar development. In *Proceedings of the Workshop on Ideas and Strategies for Multilingual Grammar Development, at the 15th European Summer School in Logic Language and Information*, Vienna, Austria, 2003. 1, 4, 5, 7, 10, 24, 103, 104, 110, 111, 112, 113, 114, 115, 116, 121, 125, 131, 133

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. Long-distance dependency resolution in automatically acquired wide-coverage pcfg-based lfg approximations. In *42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 319–326, Barcelona, Spain, 2004. 108, 110, 203

Aoife Cahill, Martin Forst, Michael Burke, Mairéad McCarthy, Ruth O'Donovan, Christian Rohrer, Josef van Genabith, and Andy Way. Treebank-based acquisition of multilingual unification grammar resources. *Journal of Research on Language and Computation; Special Issue on Shared Representations in Multilingual Grammar Engineering*, pages 247–279, 2005. 1, 4, 5, 7, 10, 103, 104, 110, 111, 112, 113, 114, 115, 125, 131, 133

Aoife Cahill, Michael Burke, Ruth O'Donovan, Stefan Riezler, Josef van Genabith, and Andy Way. Wide-coverage deep statistical parsing using automatic dependency structure annotation. *Computational Linguistics*, 34(1):81–124, 2008. 1, 105

John A. Carroll and Ted Briscoe. Apportioning development effort in a probabilistic lr parsing system through evaluation. In *Proceedings of the ACL/SIGDAT Conference on Empirical Methods in Natural Language Processing*, pages 92–100, Philadelphia, PA, 1996. 23, 30, 36

John A. Carroll, Ted Briscoe, and Antonio Sanfilippo. Parser evaluation: a survey and a new proposal. In *Proceedings of the 1st International Conference on Language Resources and Evaluation*, pages 447–454, Granada, Spain, 1998. 23, 31

Eugene Charniak. Tree-bank grammars. Technical report, Department of Computer Science, Brown University, Portland, OR, 1996. 22

Eugene Charniak and M. Johnson. Coarse-to-fine nbest-parsing and maxent discriminative reranking. In *43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pages 173–180, Ann Arbor, Michigan, 2005. 22

Eugene Charniak, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. Multilevel coarse-to-fine pcfg parsing. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL-06)*, pages 168–175, New York, NY, 2006. 22

John Chen and Vijay K. Shanker. Automated extraction of tags from the penn treebank. In *Proceedings of the 6th International Workshop on Parsing Technologies (IWPT-00)*, pages 65–76, Trento, Italy, 2000. 4

Grzegorz Chrupala, Nicolas Stroppa, Josef van Genabith, and Georgiana Dinu. Better training for function labeling. In *Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP 2007)*, pages 133–138, Borovets, Bulgaria, 2007. 142, 150, 151, 165, 207

Montserrat Civit and Ma Marti. Building cast3lb: A spanish treebank. *Research on Language and Computation*, 2(4):549–574, December 2004. 208

Stephen Clark and James R. Curran. Log-linear models for wide-coverage ccg parsing. In *Proceedings of the SIGDAT Conference on Empirical Methods in*

Natural Language Processing (EMNLP '03), pages 97–104, Sapporo, Japan, 2003. 104

Stephen Clark and James R. Curran. Parsing the wsj using ccg and log-linear models. 2004. 104

Michael Collins. Three generative, lexicalised models for statistical parsing. In *35th Annual Meeting of the Association for Computational Linguistics (ACL-97, jointly with the 8th Conference of the EACL)*, pages 16–23, Madrid, Spain, 1997. 22, 25

Anna Corazza, Alberto Lavelli, and Giorgio Satta. Measuring parsing difficulty across treebanks. Technical report, 2008. 62, 72

Richard S. Crouch, Ronald M. Kaplan, Tracy H. King, and Stefan Riezler. A comparison of evaluation metrics for a broad coverage parser. In *Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems; LREC-02 Workshop*, pages 67–74, Las Palmas, Spain, 2002. 111

Mary Dalrymple. *Lexical-Functional Grammar*. Academic Press, London, United Kingdom, 2001. 105, 108, 110

Michael Daum, Kilian Foth, and Wolfgang Menzel. Automatic transformation of phrase treebanks to dependency trees. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-04)*, pages 1149–1152, Lisbon, Portugal, 2004. 82

Stefanie Dipper. Implementing and documenting large-scale grammars — german lfg, doctoral dissertation, ims, university of stuttgart. *Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung (AIMS)*, 9(1), 2003. 8, 119, 120, 124, 199, 201

Erich Drach. *Grundgedanken der Deutschen Satzlehre*. reprint Darmstadt, Wissenschaftliche Buchgesellschaft, 1963, Diesterweg, Frankfurt/M., 1937. 15

Amit Dubey. *Statistical Parsing for German: Modeling Syntactic Properties and Annotation Differences*. PhD dissertation, Computational Linguistics, Saarland University, Saarbrücken, Germany, 2004. 30, 81

Amit Dubey. What to do when lexicalization fails: Parsing german with suffix analysis and smoothing. In *43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pages 314–321, Ann Arbor, Michigan, 2005. 29, 30

Amit Dubey and Frank Keller. Probabilistic parsing for german using sister-head dependencies. In *41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 96–103, Sapporo, Japan, 2003. 6, 14, 20, 24, 25, 26, 35

Oskar Erdmann. *Grundzüge der deutschen Syntax nach ihrer geschichtlichen Entwicklung dargestellt*. Verlag der J. G. Cotta'schen Buchhandlung, Stuttgart, 1886. 15

Sisay Fissaha, Daniel Olejnik, Ralf Kornberger, Karin Müller, and Detlef Prescher. Experiments in german treebank parsing. In *Proceedings of the 6th International Conference on Text, Speech and Dialogue (TSD-03)*, pages 50–57, Ceske Budejovice, Czech Republic, 2003. 20, 24, 25, 31

Martin Forst. Treebank conversion - creating an f-structure bank from the tiger corpus. In *Proceedings of the 8th International Lexical Functional Grammar Conference (LFG-03)*, pages 205–216, Saratoga Springs, NY, USA, 2003. 111, 114, 115, 116

Martin Forst. Filling statistics with linguistics - property design for the disambiguation of german lfg parses. In *Proceedings of the ACL Workshop on Deep Linguistic Processing*, pages 17–24, Prague, Czech Republic, 2007. 199, 201, 202

Martin Forst, Núria Bertomeu, Berthold Crysmann, Frederik Fouvry, Silvia Hansen-Schirra, and Valia Kordoni. Towards a dependency-based gold standard for german parsers - the tiger dependency bank. In *Proceedings of the COLING Workshop on Linguistically Interpreted Corpora (LINC '04)*, pages 31–38, Geneva, Switzerland, 2004. 115

Jennifer Foster, Joachim Wagner, Djamé Seddah, and Josef van Genabith. Adapting wsj-trained parsers to the british national corpus using in-domain self-

training. In *Proceedings of the 10th International Conference on Parsing Technologies (IWPT-07)*, pages 33–35, Prague, Czech Republic, 2007. 22

Kilian Foth. Eine umfassende Dependenzgrammatik des Deutschen. Technical report, Fachbereich Informatik, Universität Hamburg, Hamburg, Germany, 2003. 82

Kilian Foth, Michael Daum, and Wolfgang Menzel. A broad-coverage parser for german based on defeasible constraints. In *Proceedings of KONVENS 2004*, Vienna, Austria, 2004. 121, 125

Anette Frank. A (discourse) functional analysis of asymmetric coordination. In *Proceedings of the 7th International Lexical Functional Grammar Conference (LFG-02)*, Athens, Greece, 2002. 92

Michael Gamon, Eric Ringger, Zhu Zhang, Robert Moore, and Simon Corston-Oliver. Extraposition: a case study in german sentence realization. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1–7, Morristown, NJ, USA, 2002. 12, 84

Daniel Gildea. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 167–202, Pittsburgh, PA, USA, 2001. 22

Claire Grover, John A. Carroll, and Ted Briscoe. The alvey natural language tools grammar (4th release). Technical Report 284, University of Cambridge: Computer Laboratory, Cambridge, UK, 1993. 22

Hubert Haider. Downright down to the right. In Uli Lutz and Jürgen Pafel, editors, *On Extraction and Extraposition in German*, Linguistik Aktuell 11, pages 245–271. John Benjamins, Amsterdam, 1996. 88

Karin Harbusch and Gerard Kempen. Clausal coordinate ellipsis in german: The tiger treebank as a source of evidence. In *Proceedings of the Sixteenth Nordic Conference of Computational Linguistics (NODALIDA)*, pages 81–88, Tartu, Estonia, 2007. 85

Simon Herling. Ueber die topik der deutschen sprache. *Abhandlungen des frankfurtischen Gelehrtenvereines für deutsche Sprache*, 3:296–362, 1821. 15

Caroline Heycock and Anthony Kroch. Minimale syntax. verb movement and the status of subjects: Implications for the theory of licensing. *Groninger Arbeiten zur germanistischen Linguistik*, 36:75–102, 1993. 92

Donald Hindle and Mats Rooth. Structural ambiguity and lexical relations. *Computational Linguistics*, 19:103–120, 1993. 22, 84

Julia Hockenmaier. Parsing with generative models of predicate-argument structure. In *41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 359–366, Sapporo, Japan, 2003. 173

Julia Hockenmaier. Creating a ccgbank and a wide-coverage ccg lexicon for german. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 505–512, Sydney, Australia, 2006. 4, 111

Julia Hockenmaier and Mark Steedman. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of Third International Conference on Language Resources and Evaluation*, pages 1974–1981, Las Palmas, Canary Islands, Spain, 2002a. 4, 104, 143, 148

Julia Hockenmaier and Mark Steedman. *CCGbank: User's Manual*. Philadelphia, PA, 2005. 104

Julia Hockenmaier and Mark Steedman. Generative models for statistical parsing with combinatory categorial grammar. In *40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pages 335–342, Philadelphia, PA, 2002b. 104

Tilman Höhle. Der begriff 'mittelfeld', anmerkungen über die theorie der topologischen felder. In *Akten des Siebten internationalen Germanistenkongresses*, pages 329–340, Göttingen, Germany, 1986. 15

Tilman Höhle. Assumptions about asymmetric coordination in german. *Grammar in progress. Glow essays for Henk van Riemsdijk*, pages 221–235, 1990. 92

Thorsten Joachims. *Learning to Classify Text using Support Vector Machines (Kluwer International Series in Engineering and Computer Science)*. Springer, Berlin, 2002. 151

Mark Johnson. Pcfg models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632, 1998. 24, 26, 27, 101, 112, 113

John Judge, Michael Burke, Aoife Cahill, Ruth O'Donovan, Josef van Genabith, and Andy Way. Strong domain variation and treebank-induced lfg resources. In *Proceedings of the 10th International Lexical Functional Grammar Conference (LFG-05)*, pages 186–204, Bergen, Norway, 2005. 22

Ronald M. Kaplan and John Maxwell. Constituent coordination in lexical-functional grammar. In *Proceedings of the 12th International Conference on Computational Linguistics*, pages 303–305, Budapest, Hungary, 1988. 84

Ronald M. Kaplan and John T. Maxwell III. An algorithm for functional uncertainty. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING-88)*, pages 297–302, Budapest, Hungary, 1988. 22

Ronald M. Kaplan and Annie Zaenen. *Long-Distance Dependencies, Constituent Structure and Functional Uncertainty*, pages 17–42. Chicago University Press, 1988. 108, 110

Ronald M. Kaplan, Stefan Riezler, Tracy H. King, John. T. Maxwell III, Alexander Vasserman, and Richard Crouch. Speed and accuracy in shallow and deep stochastic parsing. In *Proceedings of the Human Language Technology Conference and the 4th Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL-04)*, pages 97–104, Boston, MA, 2004. 1, 105

Andreas Kathol. Linearization vs. phrase structure in german coordination constructions. *Cognitive Linguistics*, 4(10):303–342, 1999. 92

Adam Kilgariff. Comparing corpora. *International Journal of Corpus Linguistics*, 6(1):1–37, 2001. 66

Dan Klein and Chris Manning. Accurate unlexicalized parsing. In *41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 423–430, Sapporo, Japan, 2003. 22, 26, 29, 82, 150

Sandra Kübler. How do treebank annotation schemes influence parsing results? Or how not to compare apples and oranges. In *Proceedings of the 5th International Conference on Recent Advances in Natural Language Processing (RANLP 2005)*, pages 293–300, Borovets, Bulgaria, 2005. x, 6, 8, 20, 35, 36, 49, 69, 88, 143, 146, 182, 183, 204

Sandra Kübler. The page 2008 shared task on parsing german. In *ACL Workshop on Parsing German (PaGe-08)*, pages 55–63, Columbus, OH, 2008. 27, 31, 148

Sandra Kübler and Heike Telljohann. Towards a dependency-oriented evaluation for partial parsing. In *Beyond PARSEVAL – Towards Improved Evaluation Measures for Parsing Systems (LREC-02 Workshop)*, Canary Islands, Spain, 2002. 31, 56, 96

Sandra Kübler, Erhard W. Hinrichs, and Wolfgang Maier. Is it really that difficult to parse german? In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, EMNLP 2006*, pages 111–119, Sydney, Australia, 2006. 6, 9, 26, 27, 31, 35, 36, 48, 49, 50, 55, 59, 60, 204

Sandra Kübler, Wolfgang Maier, Ines Rehbein, and Yannick Versley. How to compare treebanks. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC-08)*, pages 2322–2329, Marrakech, Morocco, 2008. 81, 185

Sandra Kübler, Ines Rehbein, and Josef van Genabith. Tepacoc - a testsuite for testing parser performance on complex german grammatical constructions. In *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories (TLT-09)*, pages 15–28, Groningen, Netherlands, 2009. 81

Vladimir. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics*, 10:707–710, 1966. 37, 42

Dekang Lin. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1420–1427, 1995. 23, 31, 96

Dekang Lin. A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 4:1420–1427, 1998. 23, 31, 56, 96

David M. Magerman. Statistical decision-tree models for parsing. In *33rd Annual Meeting of the Association for Computational Linguistics (ACL-95)*, pages 276–283, Cambridge, MA, 1995. 22, 107, 125

Claudia Maienborn. Das zustandspassiv: Grammatische einordnung - bildungsbeschränkungen - interpretationsspielraum. *Zeitschrift für Germanistische Linguistik*, 1(35):83–114, 2007. 124

Wolfgang Maier. Annotation schemes and their influence on parsing results. In *Proceedings of the COLING-ACL-06 Student Research Workshop*, pages 19–24, Sydney, Australia, 2006. 6, 9, 20, 35, 36, 48, 49, 55, 204

Christopher Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999. 31

Mitchell Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. 1, 22

David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-06)*, pages 152–159, New York, NY, 2006a. 22

David McClosky, Eugene Charniak, and Mark Johnson. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 337–344, Sydney, Australia, 2006b. 22

Scott Miller and Heidi J. Fox. Automatic grammar acquisition. In *Proceedings of the workshop on Human Language Technology*, pages 268–271, Plainsboro, NJ, 1994. 3

Yusuke Miyao and Jun'ichi Tsujii. Probabilistic disambiguation models for wide-coverage hpsg parsing. In *43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pages 83–90, Ann Arbor, MI, 2005. 4, 104

Yusuke Miyao and Jun'ichi Tsujii. Maximum entropy estimation for feature forests. In *Proceedings of the 2nd International Conference on Human Language Technology Research*, pages 292–297, San Diego, CA, 2002. 104

Gereon Müller. On extraposition and successive cyclicity. In Robert Freidin and Howard Lasnik, editors, *Syntax. Critical Concepts in Linguistics*, volume III of *Transformations (2)*, pages 65–92. Routledge, London & New York, 2006. 88

Stefan Müller. Zur Analyse der scheinbar mehrfachen Vorfeldbesetzung. *Linguistische Berichte*, 203:297–330, 2005. http://hpsg.fu-berlin.de/~stefan/Pub/mehr-vf-lb.html. 15

Hiroko Nakanishi, Yusuke Miyao, and Jun'ichi Tsujii. Using inverse lexical rules to acquire a wide-coverage lexicalized grammar. In *IJCNLP 2004 Workshop on Beyond Shallow Analyses - Formalisms and Statistical Modeling for Deep Analyses*, Sanya City, Hainan Island, China, 2004. 4, 104

John Nerbonne and Wybo Wiersma. A measure of aggregate syntactic distance. In *Proceedings of the Workshop on Linguistic Distances, at the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL-06)*, pages 82–90, Sydney, Australia, 2006. 62

Ruth O'Donovan, Michael Burke, Aoife Cahill, Josef van Genabith, and Andy Way. Large-scale induction and evaluation of lexical resources from the penn-ii treebank. In *42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 367–374, Barcelona, Spain, 2004. 108, 109, 188

Ruth O'Donovan, Michael Burke, Aoife Cahill, Josef van Genabith, and Andy Way. Large-scale induction and evaluation of lexical resources from the penn-ii

and penn-iii treebanks. *Computational Linguistics*, 31(3):329–366, 2005a. 109, 188

Ruth O'Donovan, Aoife Cahill, Josef van Genabith, and Andy Way. Automatic acquisition of spanish lfg resources from the cast3lb treebank. In *Proceedings of the 10th International Lexical Functional Grammar Conference (LFG-05)*, pages 334–352, Bergen, Norway, 2005b. 4, 110

Stephan Oepen. Beyond the science of the wall street journal. Talk at the Unified Linguistic Annotation Workshop (ULA-07). Bergen, Norway, 2007. 22

Fernando Pereira and Yves Schabes. Inside-outside reestimation from partially bracketed corpora. In *30th Annual Meeting of the Association for Computational Linguistics (ACL-92)*, pages 128–135, Newark, DE, 1992. 3

Slav Petrov and Dan Klein. Improved inference for unlexicalized parsing. In *Proceedings of the Human Language Technology Conference and the 7th Annual Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL-07)*, pages 404–411, Rochester, NY, 2007. 22, 24, 142, 150

Slav Petrov and Dan Klein. Parsing german with language agnostic latent variable grammars. In *ACL Workshop on Parsing German (PaGe-08)*, pages 33–39, Columbus, OH, 2008. 24, 27, 31, 148, 150

Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press and CSLI Publications, Chicago, IL, 1994. 84

Anna N. Rafferty and Christopher D. Manning. Parsing three german treebanks: Lexicalized and unlexicalized baselines. In *ACL Workshop on Parsing German (PaGe-08)*, pages 40–46, Columbus, OH, 2008. 24, 29, 30

Ines Rehbein and Josef van Genabith. Evaluating evaluation measures. In *Proceedings of the 16th Nordic Conference of Computational Linguistics NODALIDA-2007*, pages 372–379, Tartu, Estonia, 2007a. 36

Ines Rehbein and Josef van Genabith. Why is it so difficult to compare treebanks? tiger and tba-d/z revisited. In *Proceedings of the 6th International Workshop*

on Treebanks and Linguistic Theories TLT-07, pages 115–126, Bergen, Norway, 2007b. 63

Ines Rehbein and Josef van Genabith. Treebank annotation schemes and parser evaluation for German. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-07)*, pages 630–639, Prague, Czech Republic, 2007c. 36

Brian Roark and Michiel Bacchiani. Supervised and unsupervised pcfg adaptation to novel domains. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL-03)*, pages 126–133, Edmonton, AB., Canada, 2003. 22

Christian Rohrer and Martin Forst. Improving coverage and parsing quality of a large-scale lfg for german. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC-06)*, pages 2206–2211, Genoa, Italy, 2006. 8, 199, 201, 207

Ivan A. Sag, Gerald Gazdar, Thomas Wasow, and Steven Weisler. Coordination and how to distinguish categories. Technical report, CSLI-84-3. Center for the Study of Language and Information, Stanford, CA, 1984. 84

Geoffrey Sampson. A proposal for improving the measurement of parse accuracy. *International Journal of Corpus Linguistics*, 5(1):53–68, 2000. 36

Geoffrey Sampson and Anna Babarczy. A test of the leaf-ancestor metric for parse accuracy. *Journal of Natural Language Engineering*, 9:365–380, 2003. 23, 31, 36, 42, 70

Geoffrey Sampson, Robin Haigh, and Eric Atwell. Natural language analysis by stochastic optimization: a progress report on project april. *Journal of Experimental and Theoretical Artificial Intelligence*, 1:271–287, 1989. 22, 31

Nathan C. Sanders. Measuring syntactic differences in british english. In *Proceedings of the COLING-ACL-07 Student Research Workshop*, pages 1–6, Prague, Czech Republic, 2007. 62

Michael Schiehlen. Annotation strategies for probabilistic parsing in german. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, pages 390–396, Geneva, Switzerland, 2004. 6, 20, 24, 26, 27, 28, 29, 30

Anne Schiller, Simone Teufel, and Christine Thielen. Guidelines für das tagging deutscher textkorpora mit stts. Technical report, Universität Stuttgart and Universität Tübingen, Tübingen, Germany, 1995. 14

Helmut Schmid. LoPar: Design and implementation. Technical report, Universität Stuttgart, Stuttgart, Germany, 2000. 24, 35, 59, 82

Helmut Schmid. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04)*, pages 162–168, Geneva, Switzerland, 2004. 50, 69, 82, 95, 112, 150

Satoshi Sekine and Michael Collins. Evalb - bracket scoring program. Retrievable from: http://cs.nyu.edu/cs/projects/proteus/evalb/, 1997. 24

Robert Sharman, Fred Jelinek, and Robert Mercer. Generating a grammar for statistical training. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 267–274, Hidden Valley, PA, 1990. 3, 22

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. An annotation scheme for free word order languages. In *Proceedings of the 5th Applied Natural Language Processing Conference (ANLP-97)*, pages 88–95, Washington, D.C., 1997. 4, 13, 21, 24, 125, 133

Mark Steedman. Dependency and coordination in the grammar of dutch and english. *Language*, 61:523–568, 1985. 84

Mark Steedman. Gapping as constituent coordination. *Linguistics and Philosophy*, 13:207–263, 1990. 92

Heike Telljohann, Erhard W. Hinrichs, Sandra Kübler, and Heike Zinsmeister. *Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)*. Universität Tübingen, Germany, 2005. 14

Josef Van Genabith, Louisa Sadler, and Andy Way. Data-driven compilation of lfg semantic forms. In *Workshop on Linguistically Interpreted Corpora (LINC-99)*, pages 69–76, Bergen, Norway, 1999. 109

Yannick Versley. Parser evaluation across text types. In *Proceedings of the 4th Workshop on Treebanks and Linguistic Theories (TLT-05)*, pages 209–220, Barcelona, Spain, 2005. 20, 22, 24, 28, 29, 82, 121

Yannick Versley and Heike Zinsmeister. From surface dependencies towards deeper semantic representations. In *Proceedings of the 5th Workshop on Treebanks and Linguistic Theories (TLT-06)*, pages 115–126, Prague, Czech Republic, 2006. 116

Dieter Wunderlich. Some problems of coordination in german. *Natural language parsing and linguistic theories*, 4(4):289–316, 1988. 92

Fei Xia. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, Bejing, China, 1999. 4

Naiwen Xue, Fei Xia, Fu-dong Chiou, and Marta Palmer. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238, 2005. 208

Appendix: Example Trees for Five Grammatical Constructions in TePaCoC

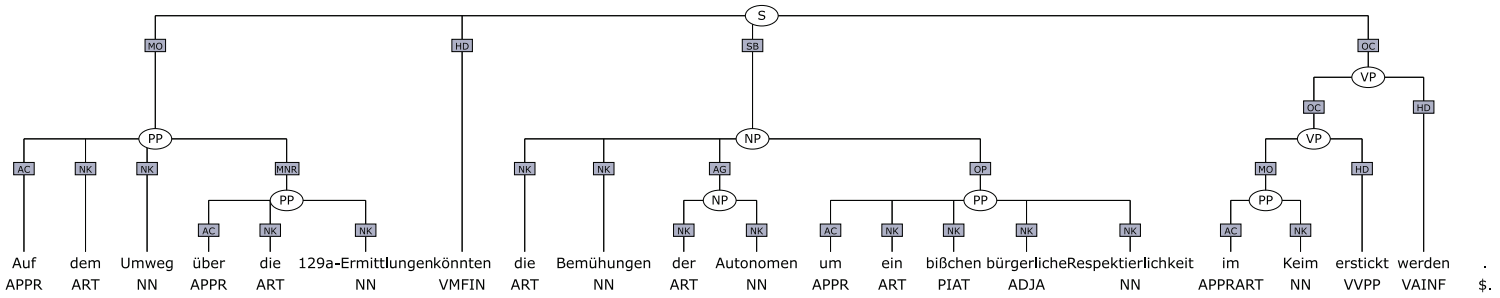


Figure 1: PP Attachment in TiGer

(40) Auf dem Umweg **über die 129a-Ermittlungen** könnten die Bemühungen der Autonomen um ein
By the detour via the 129a-investigations could the efforts of the autonomous activists for a
bißchen bürgerliche Respektierlichkeit **im Keim** erstickt werden.
little middle-class respectability in the bud nipped be.

"With the 129a investigations, the efforts of the autonomous activists for a little middle-class respectability could be nipped in the bud."

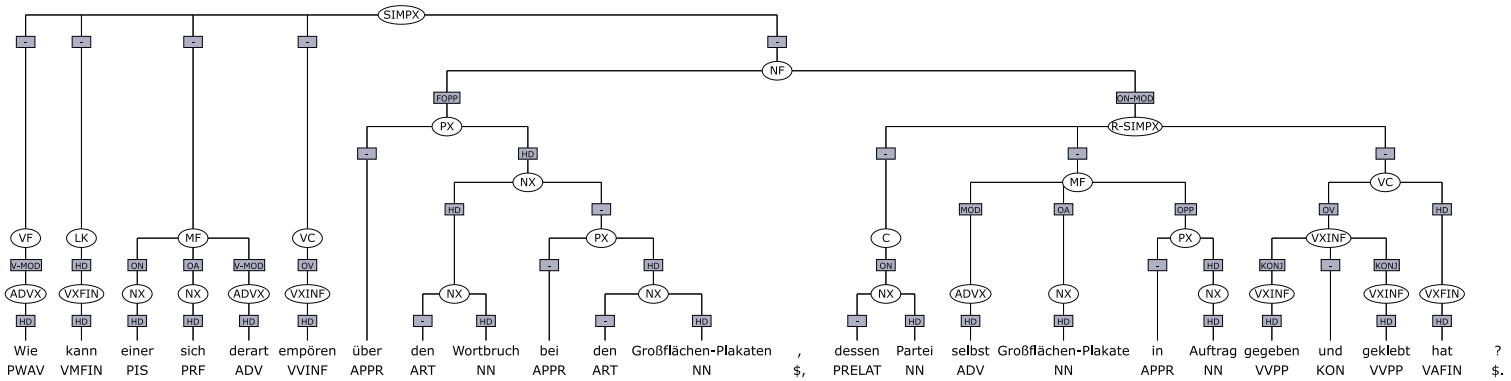


Figure 2: PP Attachment in TüBa-D/Z

(41) Wie kann einer sich derart empören über den Wortbruch **bei den Großflächen-Plakaten**,
How can one *refl.* so revolt about the breach of promise concerning the large-scale posters,
dessen Partei selbst Großflächen-Plakate **in Auftrag gegeben** und geklebt hat?
whose party itself large-scale posters in commission given and posted has?
"How can someone bristle at the breach of promise concerning the large-scale posters when his party has commissioned and posted such posters?"

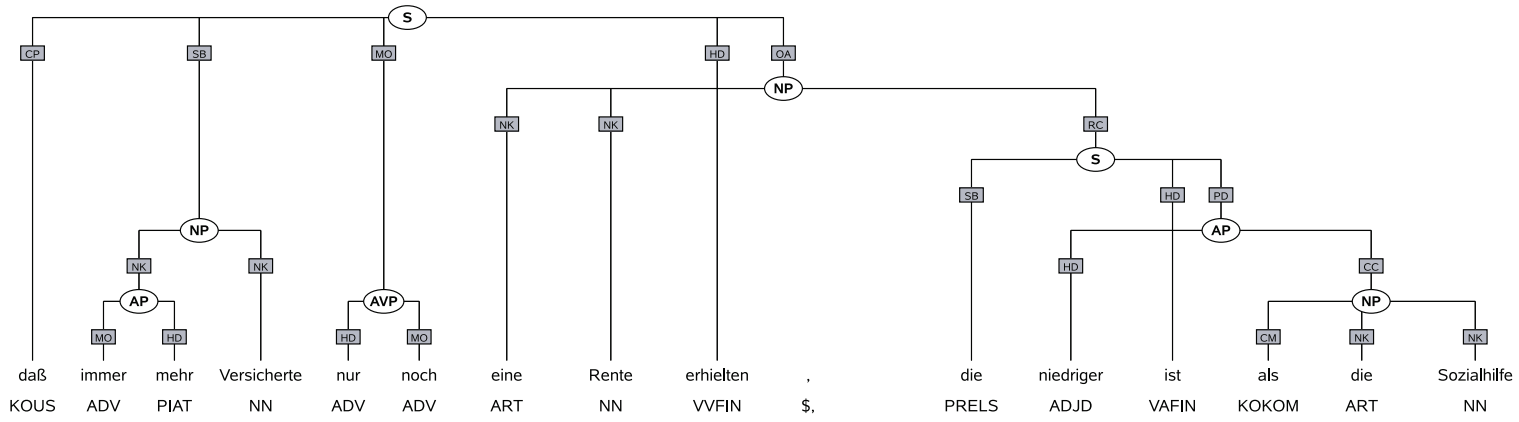


Figure 3: Extraposed Relative Clauses in TiGer

(42) ...da immer mehr Versicherte nur noch eine **Rente** erhielten, **die niedriger ist als die Sozialhilfe**
...that always more insurants just still a pension would receive, which lower is than the social welfare
"... that more and more insurants receive a pension lower than social welfare"

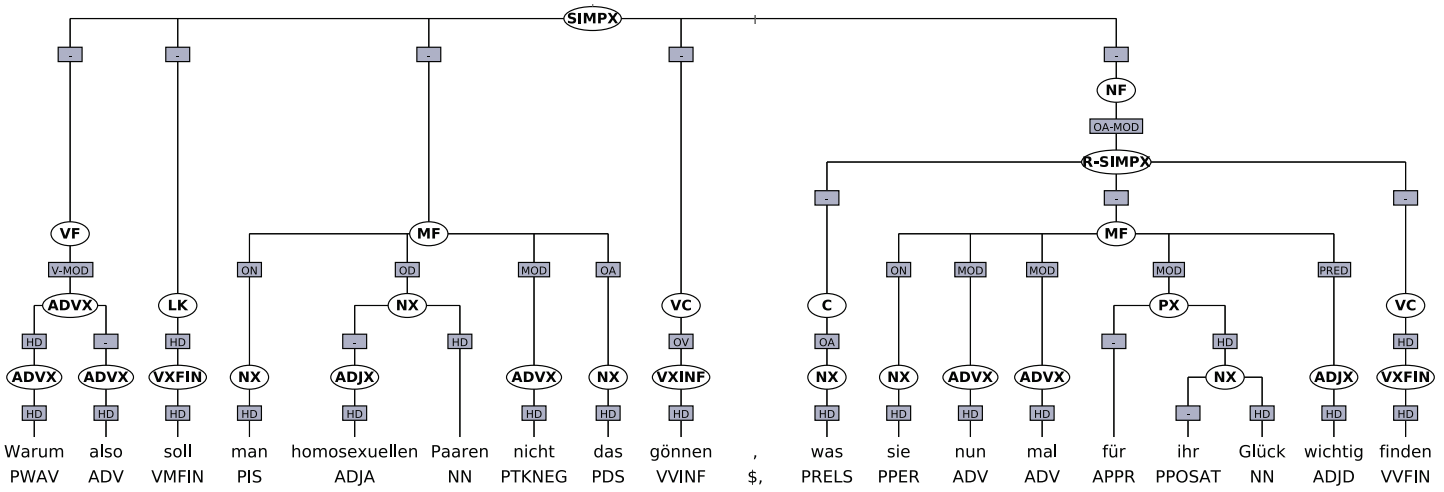


Figure 4: Extraposed Relative Clauses in TüBa-D/Z

(43) Warum also soll man homosexuellen Paaren nicht **das** gönnen, **was sie nun mal für ihr Glück wichtig**
Why so shall one homosexual couples not that grant, which they now for their luck important
finden?
find?

"So why shouldn't homosexual couples be granted what they think is important to their happiness."

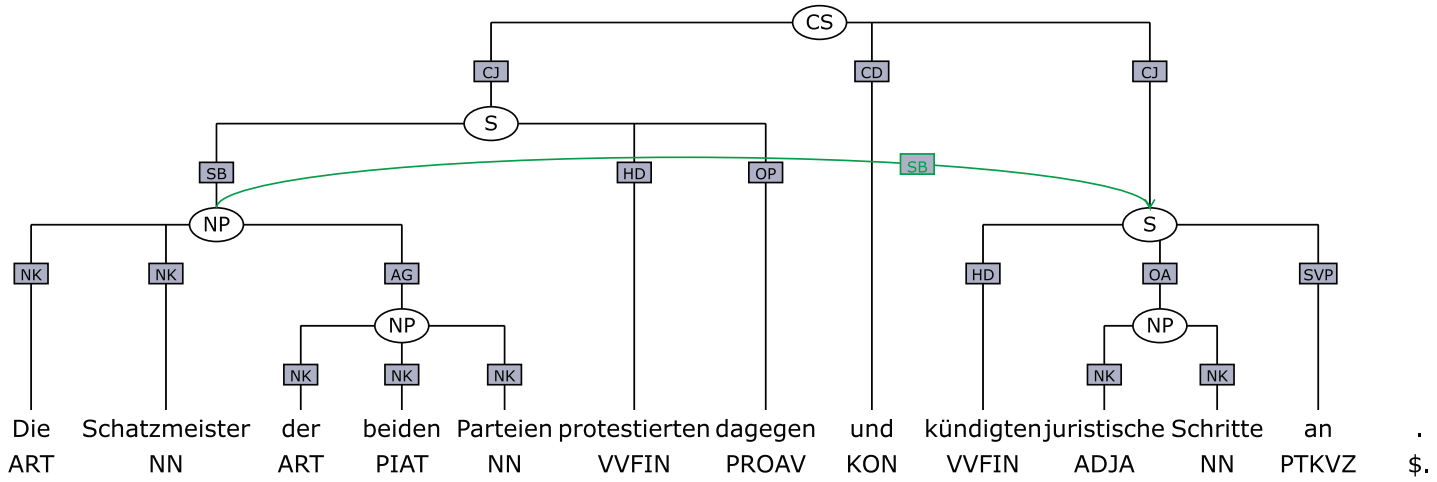


Figure 5: Forward Conjunction Reduction in TiGer

(44) Die Schatzmeister der beiden Parteien protestierten dagegen und kündigten juristische Schritte an.
The treasurers of the both parties protested against it and announced legal action verb part.
"The treasurers of both parties protested and announced they would take legal action."

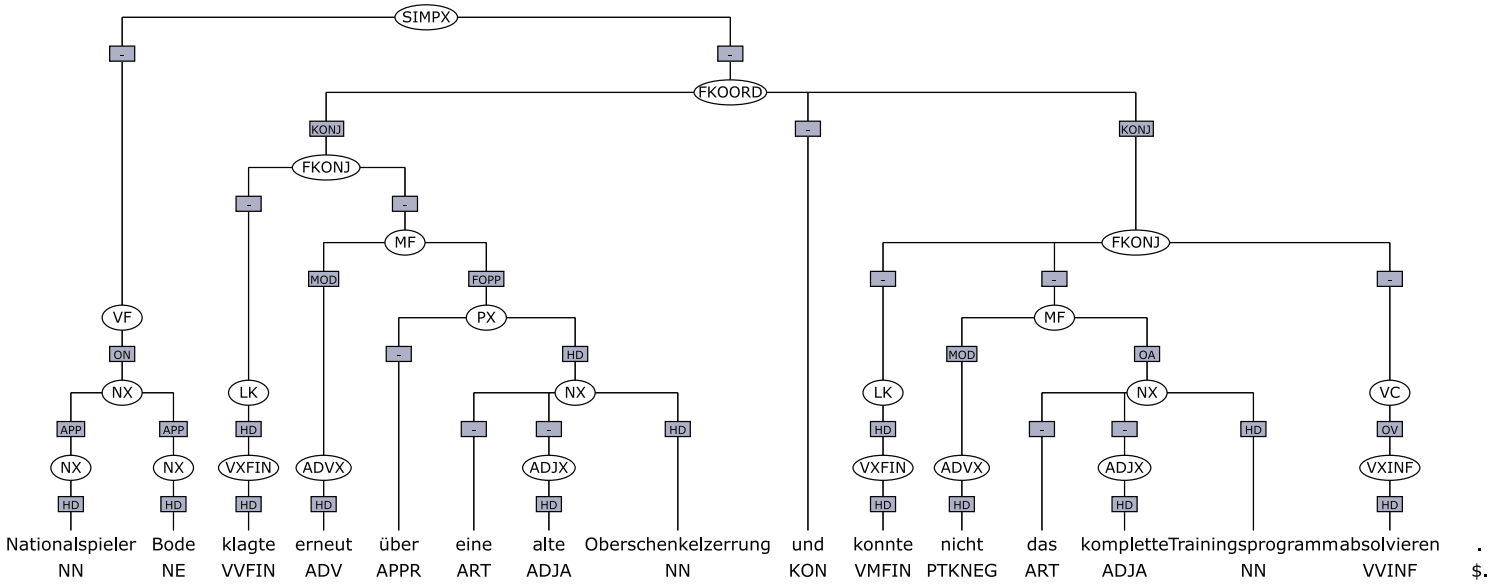


Figure 6: Forward Conjunction Reduction in TüBa-D/Z

(45) Nationalspieler Bode klagte erneut über eine alte Oberschenkelzerrung und konnte nicht das
Member of the national team Bode complained again about an old strain of the thigh and could not the
komplette Trainingsprogramm absolvieren.
complete training regime finish.

"Member of the national team Bode again complained about a strain of the femural muscle and could not finish the training."

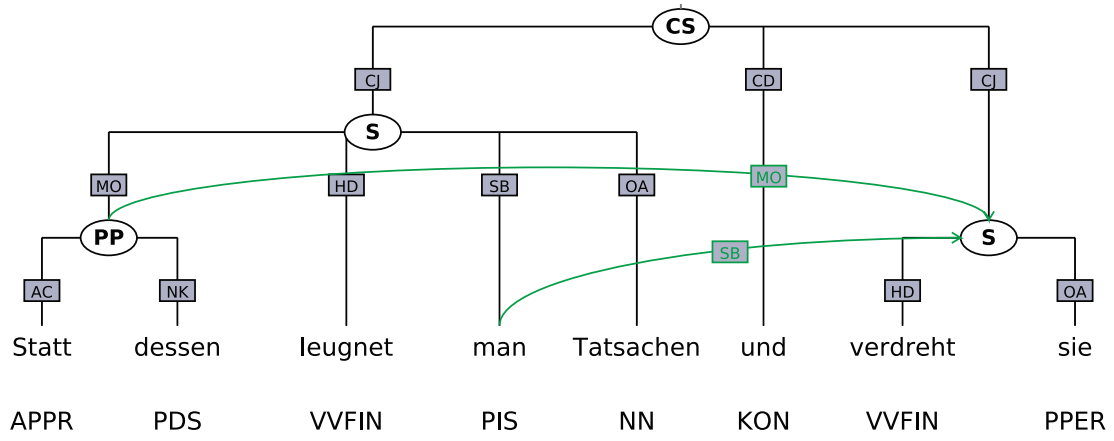


Figure 7: Subject Gap with Fronted/Finite Verbs in TiGer

(46) Statt dessen leugnet **man** Tatsachen und verdreht sie.
Instead denies one facts and twists them.
"Instead, the facts are denied and twisted."

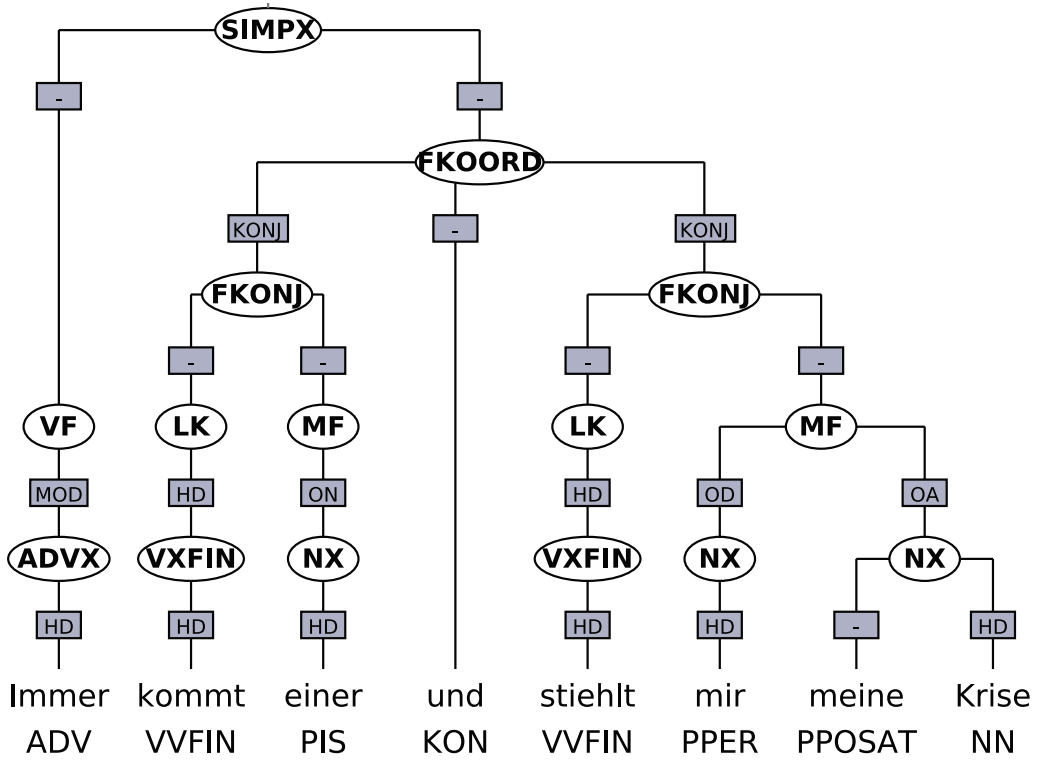


Figure 8: Subject Gap with Fronted/Finite Verbs in TüBa-D/Z

(47) Immer kommt **einer** und stiehlt mir meine Krise.
Always comes someone and steals me my crisis.
"Every time, someone comes and steals my crisis."

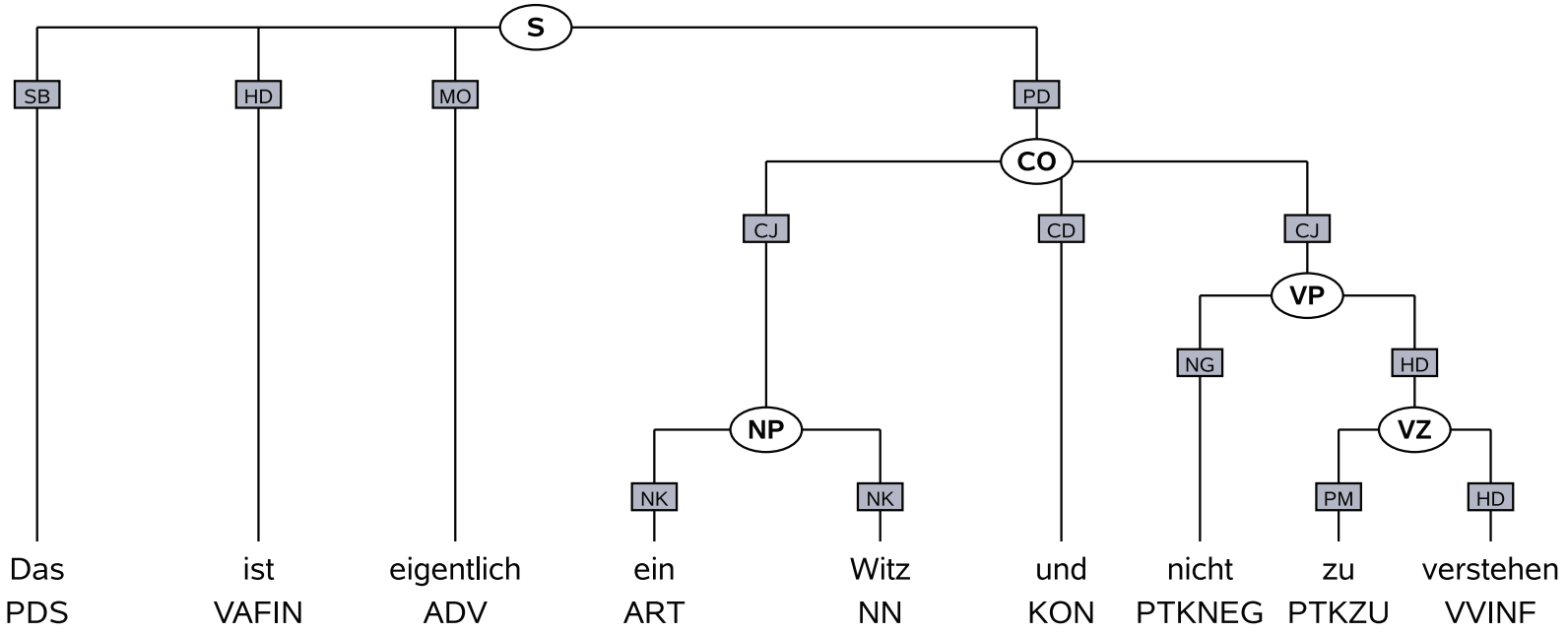


Figure 9: Coordination of Unlike Constituents in TiGer

(48) Das ist eigentlich ein Witz und nicht zu verstehen.
This is actually a joke and not to understand.
"This actually is a joke and hard to understand."

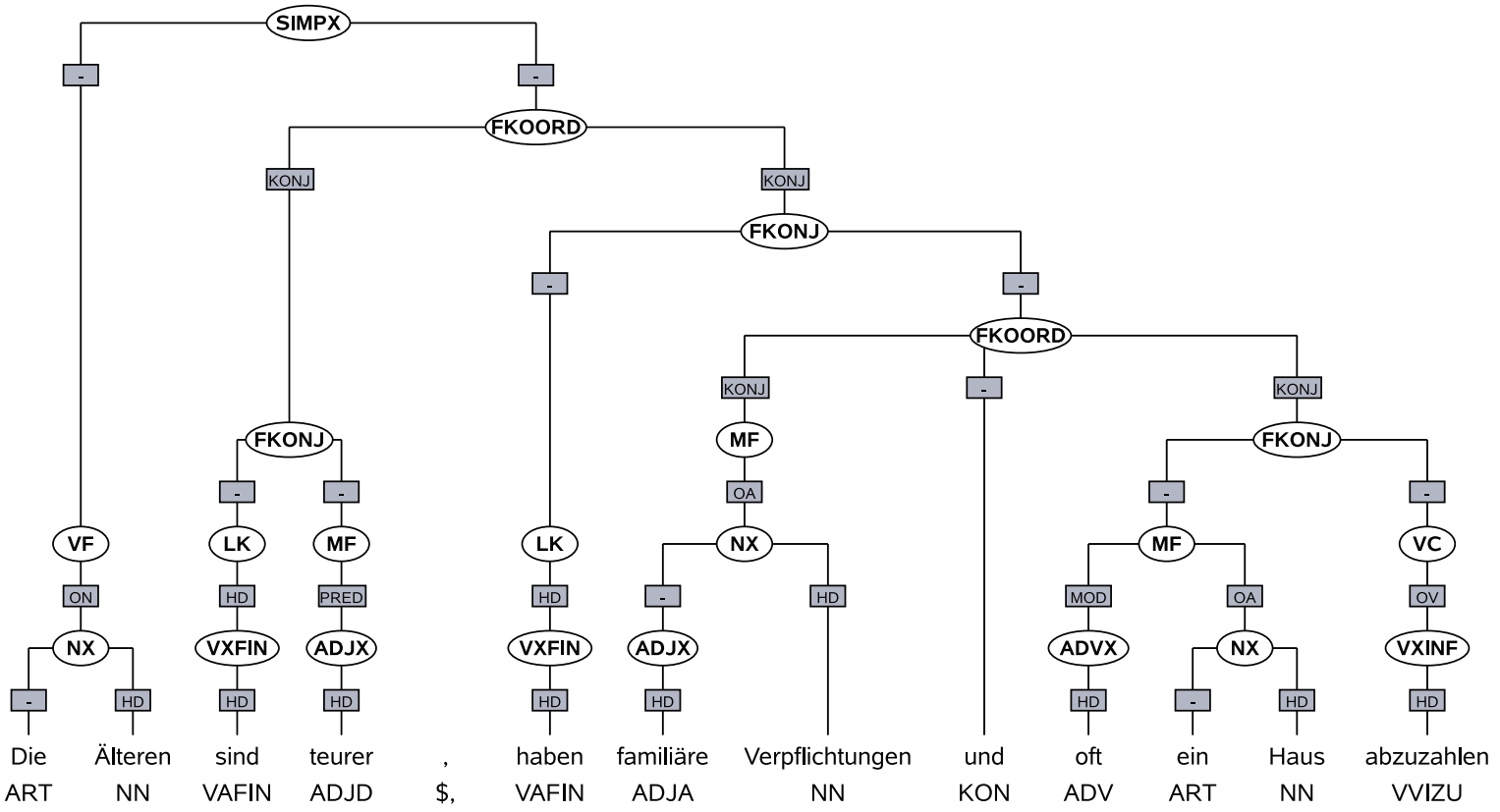


Figure 10: Coordination of Unlike Constituents in TüBa-D/Z

(49) Die Älteren sind teurer, haben familiäre Verpflichtungen und oft ein Haus abzuzahlen.
The elderly are more expensive, have familial commitments and often a house to repay.
"The elderly are more expensive, have family commitments and often have to pay off a house."